TARTU ÜLIKOOL

HUMANITAARTEADUSTE JA KUNSTIDE VALDKOND

EESTI JA ÜLDKEELETEADUSE INSTITUUT

Helen Kaljumäe

KORPUSE ANALÜÜSI PROGRAMMIDE TESTIMINE EESTIKEELSE KORPUSE PÕHJAL

Bakalaureusetöö

Juhendajad Kadri Muischnek ja Kristel Uiboaed

TARTU 2016

Sisukord

Sissejuhatus

Palju keeleteaduslikke uurimistöid põhineb praegu tekstikorpuse analüüsil. Eesti keele jaoks on olemas suured korpused oma kasutajaliidestega. Olemasolevad korpused ei kata aga kõiki vajadusi ja nii võib keeleuurijal tekkida tarvidus koostada oma väike tekstikorpus ja analüüsida just seda.

Käesoleva bakalaureusetöö eesmärgiks on korpuse analüüsimise programmide testimine ja võrdlemine, pöörates eraldi tähelepanu nende sobivusele eesti keele jaoks. Töö on praktilise suunitlusega, sest eesmärk on testimise teel välja selgitada, millised programmid sobivad kasutamiseks keeleuurijale, kellel on tarvis ise luua väike korpus ning seda analüüsida. Testimisel arvestan eestikeelse teksti eripäradega ning esitatud juhiseid järgides on võimalik luua oma korpus ning seda edukalt analüüsida. Töös testin ka internetist korpuse koostamise programme, mille põhjal valmib vajalik testkorpus.

Laurence Anthony (2013: 141) ütleb oma artiklis „A critical look at software tools in corpus linguistics“, et korpust uurides ja analüüsides on oluline kasutada selleks mõeldud analüüsivahendeid, saamaks keeleandmetest kätte vajalikku informatsiooni. See, millist tööriista kasutatakse, määrab ära võimalikud uurimismeetodid. Korpuse analüüsimiseks programmi valides tuleb silmas pidada uurija vajadusi.

Korpuse analüüsi programmide võrdlemine ja tulemuste dokumenteerimine on oluline, sest erinevaid programme tekstikogude analüüsimiseks on palju ning vajadusel nende hulgast sobivaima leidmine ilma eeltööta on keeruline. Käesolev töö lihtsustab korpuse analüüsimise protsessi, sest annab juhised, millist programmi kasutada, mida ja kuidas sellega analüüsida ning millele programmi kasutamisel täpsemalt tähelepanu pöörata. Ühtlasi annab töö ülevaate olemasolevatest analüüsivahenditest ja nende sobilikkusest eestikeelse korpusega töötamisel. Töö tõstab ka üldist teadlikkust selliste programmide olemasolust ning nende kasutusvõimalustest.

Bakalaureusetöö koosneb neljast peatükist. Esimene peatükk tutvustab korpuse analüüsi programmide arengut ja seda, kuidas neid programme on varem uuritud. Teises peatükis annan ülevaate, mille alusel programmid testimiseks välja valisin ning töös kasutatud analüüsimeetoditest. Töö kolmandas peatükis testin korpuse koostamise programme, mida kasutasin ka analüüsi programmide jaoks vajaliku testkorpuse koostamiseks. Viimane, neljas peatükk esitab programmide testimise tulemused.

1. Korpus ja selle analüüsi programmid

Kennedy (1998) järgi on keeleteaduses korpus kirjalike tekstide või transkribeeritud suulise kõne kogu, mille abil on võimalik keelt uurida ja kirjeldada. Tänapäeval on olemas juba väga suuri, rohkem kui 100 miljonist sõnast koosnevaid elektroonilisi keelekorpuseid (Gries, Newman 2013: 259). Võrreldes väikeste korpustega tuleb suuremates korpustes keelele iseloomulikke jooni rohkem esile ning ülevaade on põhjalikum (Anthony 2013: 146). Korpuse suuruse puhul on olulisim arvestada, et see oleks vastavuses uurimuse eesmärgiga. Seetõttu on ka väikesed korpused oluliseks uurimisallikaks (Gries, Newman 2013: 259). Scott (2012: 82) on väitnud, et näiteks isiklikuks tarbeks loodavad korpused võivad muutuda elektrooniliste tekstimaterjalide kättesaadavuse tõttu tavapäraseks.

1.1. Korpuse analüüsi programmide areng

Korpuse analüüsimiseks on loodud erinevaid programme. Suuri korpusi on programmi abiga kergem läbi töötada kui käsitsi ning tulemus on kvaliteetsem. Programmide abil on võimalik rakendada nii kvalitatiivset kui kvantitatiivset uurimisviisi, koostades konkordantside loendeid ja sagedusandmestikke. (McEnery, Hardie 2012: 2, 37) Konkordants on sõnavorm koos kontekstiga (McEnery, Hardie 2012: 241). Milline on analüüsi lõpptulemus, oleneb kasutatavast vahendist ja selle pakutavatest võimalustest (Anthony 2013: 143). Anthony (2013: 149–150) on teinud katse näitamaks, kuidas sõnaloendid võivad erineda, sest programmid defineerivad sõna erinevalt. Sõnaloend[1] näitab, mis sõnad millise sagedusega korpuses esinevad (Bianchi 2012: 45). Eesti keeles

[1] Käesolevas töös kasutan mõistet *sõnaloend*, sest kuigi ingliskeelses kirjanduses kasutatakse ka terminit *sagedusloend*, on eesti keeles selle tähendusala kitsam.

võivad tekkida erinevused sõnaloendis näiteks sidekriipsu kasutusel. *Samm-sammult liikuma* võib programm lugeda *samm-sammult* üheks sõnaks, kui sidekriipsu pidada sõna osaks, või kaheks sõnaks ning *samm* ja *sammult* loetakse eraldi sõnavormideks.

Ingliskeelses kirjanduses tähistatakse sageli erinevaid korpuse analüüsi programme sõnaga *concordancer* ehk *konkordantsiprogramm* (nt McEnery ja Hardie 2012, Wiechmann ja Fuhs 2006), kuid kuna sellised programmid võimaldavad tihti ka muid analüüsimeetodeid peale konkordantsi väljastamise, siis kasutan töös edaspidi nimetust *korpuse analüüsi programmid*. Korpuse analüüsi programmide areng algas 20. sajandi keskpaigas, mil valmis esimene elektrooniline korpus ning mõisteti, et konkordantsi on võimalik luua edukalt ka elektroonilise tekstikogumi põhjal. Arenesid esimese generatsiooni korpuse analüüsi programmid, mis töötasid serveris. Peamiselt võimaldasid need analüüsida korpust KWIC (*keyword in context)* konkordantsi abil, muu andmetöötluse jaoks oli vaja teisi programme. Piiratud oli erinevate tähemärkide kasutamine, näiteks muudeti *é e*-ks. Puudusid standardid, sest erinevad uurimisrühmad koostasid programme ise – valmis kirjutatud programmide jagamine serverite vahel oli keeruline. (McEnery, Hardie 2012: 37–39)

1980.–1990. aastatel hakkas tehnika arengu tõttu levima korpuspõhine keeleuuurimine ning ilmusid teise generatsiooni korpuse analüüsi programmid, mis olid kasutatavad juba personaalarvutites. See muutis korpuste ja konkordantside kasutamise lihtsamaks ning võimaldas keskenduda rohkem olemasolevate programmide arendamisele. Teise generatsiooni programmid esimestest sisu poolest ei erinenud. Paraku oli märgendamine probleemsem kui esimese generatsiooni programmidega: kui enne olid töögruppidel enda koostatud korpuse analüüsi programmid, mis töötasid nende valitud vormingu järgi, siis teise generatsiooni programmide jaoks polnud märgendused sõnadest eraldatavad. Selleaegsete personaalarvutite väike mälumaht ei võimaldanud ka suurte korpuste kasutamist. (McEnery, Hardie 2012: 39–40)

Kolmanda generatsiooni korpuse analüüsi programme uuendatakse ja kasutatakse siiani. Neil on tekstianalüüsiks lisaks konkordantsile rohkem põhitööriistu, näiteks sagedusloend ja kollokatsioon. Kollokatsioon arvutilingvistikas on sõnade koos

esinemine: kollokatsiooni moodustavad sõnad, mis esinevad üksteise naabruses sagedamini, kui võiks eeldada nende eraldiesinemise sageduste põhjal (McEnery, Hardie 2012: 240). Kolmanda generatsiooni programmide andmemaht on suurem ning paljudel programmidel on olemas ka Unicode'i tugi, mis võimaldab erinevates kirjasüsteemides tekstide kasutamist. Üldiseks teksti märgendamise standardiks on kujunenud XML-märgistuskeel. (McEnery, Hardie 2012: 40–41)

Neljanda generatsiooni programmid on suunatud lahendama eelkõige eelmise generatsiooni programmidega tekkinud probleeme, näiteks arvutite võimsus ja autoriõiguste küsimused. Neljanda põlvkonna programme iseloomustab see, et need on kasutatavad veebikeskkonnas. Sellisel juhul näeb korpuse kasutaja terviktekstide asemel vaid osa korpustest, kuigi päringuid saab teha kogu korpuse ulatuses, ning õiguslikud piirangud pole enam põhjendatud. Veebiliidese positiivseks omaduseks on ka kiirus ja sõltumatus erinevatest operatsioonisüsteemidest. (McEnery, Hardie 2012: 43–44, 46)

Veebiprogrammid on kasulikud eelkõige suurte korpuste analüüsimiseks, kuid paljud neist ei võimalda kasutajal üles laadida isiklikku tekstikogu, võimalik on kasutada vaid programmiga seotud korpust. Keeruline on ka delikaatse informatsiooniga töötamine ning kasutajal puudub algteksti nägemise võimalus. (Anthony 2013: 153)

Korpuse analüüsimiseks leidub programme, mis erinevad nii funktsioonidelt kui kasutusmugavuselt (Anthony 2013: 154), kuid näiteks Gries (2009: 1235) ja Biber jt (1998: 255) on toonud välja programmeerimiskeelte oskuse olulisuse. See võimaldab lingvistil koostada programm vastavalt oma uurimuse eesmärkidele. Ise programmeerides saab luua tööriista, millega on võimalik kasutada analüüsimeeotodeid, mida valmisprogramm ei võimalda. Korpuse mahul selle puhul piiranguid ei ole ning analüüsi väljundeid on samuti erinevaid, nende hulgas ka näiteks graafiline. (Biber jt 1998: 255–256, Gries 2009: 1235–1236)

Anthony (2013: 158–159) toob välja, mis suunas võiksid areneda järgmise põlvkonna programmid. Need võiksid olla avatud lähtekoodiga, kirjutatud moodulitena ning nende loomisesse võiks kaasata erinevate oskustega inimesi. See tagaks paremini programmide laiema kasutusala.

1.2. Nõuded programmide funktsionaalsusele

Korpuse analüüsi programme on võrreldud ja testitud varemgi ning võrdluskriteeriumid on sageli erinevad. Wiechmann ja Fuhs (2006: 109–110) on analüüsinud programme kolme üldise omaduse alusel: funktsionaalsus (mida programmid teha oskavad), esitus (kiirus ja kui suure andmehulgaga need töötada suudavad) ja kasutajasõbralikkus. Lisaks programmi funktsioonidele analüüsitakse sisend-väljund formaate, kui täpset päringut on võimalik koostada, kuidas päringutulemust on võimalik sorteerida ning esitada ja kas märgendusi on võimalik peita.

Lisaks on koostatud täpsemad alakriteeriumid. Uuritakse, kui paindlik on programm materjali laadimisel (milline võib olla näiteks tähemärkide kodeering) ning kuidas sisestatud andmeid on võimalik kasutada. Samuti on alakriteeriumid erinevate tööriistade testimiseks: mil viisil saab leida andmetes esinevaid mustreid, kas on võimalik kasutada metamärke ja regulaaravaldisi ning kas saab koostada kollokatsioone ja n-gramme. (Wiechmann, Fuhs 2006: 110) Regulaaravaldis on sümbolite järjend, mis kirjeldab mingit märgijada (nt sõna või fraasi) tekstis, mis kas vastab avaldisele või mitte. N-grammide funktsiooni kasutamisel saab leida üksteisele vahetult järgnevatest sõnadest moodustunud fraase (Cheng 2011: 102).

McEnery ja Hardie (2012: 22–23) on kirjutanud korpuse analüüsi programmi kasutaja jaoks suunavad küsimused, millest juhindudes on sobivat programmi lihtsam leida. Esmalt soovitatakse uurida, kuidas saab oma korpust programmi laadida ning vajadusel vahetada. Tähelepanu tuleb pöörata sellele, millisel kujul loodud korpus olema peaks: kas materjal võib olla ühes või mitmes tekstifailis ning millises formaadis. Seejärel soovitatakse testida programmi tööd: kuidas saab teostada otsingut ühe kindla sõna leidmiseks, kas päringut on võimalik teha märgenduste järgi, kas programm eristab suur- ja väiketähti (ja kas seda on võimalik muuta) ning kas päringutulemuste hulka saab vähendada. Analüüsifunktsioonide puhul tuleks vaadata, kas sõnaloendit saab koostada sõnadest/märgenditest ja kas korpuse kohta on võimalik saada üldstatistilisi andmeid, kas on võimalik koostada kollokatsioone, n-gramme, teha märksõna analüüsi ning kuidas tulemusi salvestada saab.

Gries ja Newman (2013: 278–279) peavad oluliseks, et programm suudaks avada mitut faili korraga, toetada erinevaid kirjasüsteeme ning arvutada nii sõnaosade, sõnade kui ka sõnaliikide sagedust. Konkordantsi ja ka kollokatsiooni kasutamisel võiks programm esitada tulemused vastavalt otsitud mustrile ning esitatud ridade pikkus olla muudetav. Programm peaks suutma mõõta sõnadevahelist seost ja esitada n-gramme ning võimaldama tulemuste salvestamist ja eksportimist.

Anagnostou ja Weir (2006: 91) on kirjutanud ülevaate kollokatsioonide tuvastajatest ning nemad on pidanud oluliseks, et programm toetaks XML-formaadis faile ja võimaldaks kasutada mitut faili korraga.

Programme on analüüsinud ka Rayson (2002: 79–84) oma doktoritöös. Tema vaatleb, kas need on tasuta kasutatavad või mitte, millise operatsioonisüsteemiga töötavad, missugused teksti kodeerimise formaadid on võimalikud ning kas enne programmi kasutamist on vaja korpus indekseerida. Oluliseks peab ta ka seda, kas programm suudab märgendatud tekstiga töötada. Märgendatud korpuse puhul peaks programm vahet tegema tekstil ja märgendusel ning märgendusi peaks olema võimalik ka vastavalt vajadusele näha või peita. Funktsioonidest peab Rayson oluliseks sõnaloendite koostamise võimalust ja nende omavahelist võrdlemist, konkordantsi (ja selle sorteerimist vastavalt vajadusele) ning kollokatsiooni. Kriteeriumite hulgas on ka see, et programmi kasutades oleks võimalik valida, millise alakorpusega töötada.

2. Materjal ja metoodika

Töös uurin korpuse koostamise ja analüüsimise programme, mis puhul on peamiseks meetodiks programmide testimine ning katsetulemuste põhjal nende võrdlemine. Põhieesmärgiks on välja selgitada, millised programmid sobivad oma väikese korpuse analüüsimiseks.

2.1. Korpuse koostamine

Internetist tekste kogudes on võimalik korpus koostada automaatselt selleks otstarbeks loodud programmidega, mida olen töös testinud. Programmide valikul lähtusin sellest, et programm annaks valitud tekstidest tulemuseks tervikliku korpuse. Testimist alustasin programmiga BootCaT (BootCaT) ning selle käigus lisandusid WebBootCat (WebBootCat) ning TextSTAT (TextSTAT).

Programme analüüsin selle põhjal, kas ja kuidas nad eri stiilis tekste internetilehekülgedelt koguda suudavad: kas valminud korpus sisaldab valitud tekste täies mahus ning kui palju on kõrvalist teksti. Korpuse loomiseks vajalikud internetileheküljed valisin jaanuarikuu jooksul Postimehe majandusuudiste rubriigist (38 uudislugu, kokku 9036 sõna)[2] ning loomateemalistest foorumitest (27 teemat, kokku 19 117 sõna)[3] eesmärgiga, et esindatud oleks korrektne kirjakeel ning vabam keele-kasutus.

[2] majandus24.postimees.ee

[3] http://kodukauniks.postimees.ee; http://www.pisi.ee; http://naistekas.delfi.ee/foorum; http://www.eeva.ee/foorum; http://www.perefoorum.ee

2.2. Korpuse analüüsi programmid

Ülevaate saamiseks uurisin esmalt internetis võimalikult paljude programmide kohta. Informatsiooni programmidest sain nende kodulehekülgedelt ning dokumentatsioonidest (täpne loetelu on leitav kirjanduse loendist). Valiku tegemiseks, milliseid neist töös põhjalikumalt testida, koostasin esmased kriteeriumid (vt tabel 1). Olulisim oli, et programmid oleksid tasuta kasutatavad, sest töö sihtrühmaks on tavakasutajad, ning toetaksid Unicode'i kodeeringut, mis on vajalik eesti keele tähestikuga töötamisel. Töös testin Windowsi operatsioonisüsteemil töötavaid programme. Läbi vaadatud programmide seast on tabelisse 1 lisamata jäetud tasulised programmid Wmatrix, Sketch Engine, Monoconc Pro, Collocate, Paraconc ning Concordance.

Kriteeriumitele vastavuse põhjal valisin töösse kuus programmi (Simple Concordance Program, MonoconcEsy, AntConc, Multilingual Corpus Toolkit, TextSTAT, WordSmith Tool), mille puhul võis olla kindel, et need töötavad vähemalt Windowsiga ning toetavad eesti tähestiku jaoks sobivat tähekodeeringut. WordSmith Tooli testimisel selgus, et prooviversioon piirab näidatavate tulemuste hulka 50-le ja 100-le, mistõttu loobusin selle programmi edasisest analüüsimisest.

Tabel 1. **Ülevaade uuritud korpuse analüüsi programmidest.**

	Tasuta kasuta-tav	Kodeering	Operatsiooni-süsteem	Funktsioonid	Muu info
AntConc	JAH	Unicode/ UTF-8	Win/Mac/Linux	Konkordants, n-grammid, sõnaloend, märksõnaloend	-
NoSketch Engine	JAH	Unicode/ UTF-8	Linux (Win-ga ei pruugi töötada)	Konkordants, sõnaloend, kollokatsioon	-
Xaira	JAH	Unicode	Erinevad operatsiooni-süsteemid	Päringud, kollokatsioon jm	XML-märgendusega korpuse jaoks. Suure korpuse jaoks
Multilingual Corpus Toolkit	JAH	Unicode	Win	Konkordants, n-grammid, kollokatsioon	Ka internetist korpuse kogumise funktsioon
ConcApp	JAH	Info puudub	Info puudub	Konkordants, sõnaloend jm	Väga vähe infot, alla laadimise link ei tööta
CQPweb	JAH	Unicode/ UTF-8	Mac/Unix	Päringud, kollokatsioon, sõnaloend jm	Suure korpuse jaoks
TextSTAT	JAH	Unicode	Win/Mac/Linux	Konkordants, sõnaloend	Ka internetist korpuse kogumise funktsioon
PowerConc	JAH	ANSI	Win	Konkordants, sõnaloend, märksõnaloend jm	-
adTAT	JAH	Info puudub	Win/Mac	Konkordants, sõnaloend, kollokatsioon jm	-
KH Coder	JAH	-	Win/Mac/Linux	Konkordants, sõnaloend, kollokatsioon jm	Kindlad keeled, millega töötada saab (eesti keelt nende hulgas pole)
LEXA	JAH	ASCII	Info puudub	Sõnaloend, päringud jm	-
MonoconcEsy	JAH	Unicode	Win	Konkordants, sõnaloend, kollokatsioon	Märksõnaloend puudub
Simple Concordance Program	JAH	Unicode	Win/Mac	Konkordants, sõnaloend, statistika	-
WordSmith Tool	EI	Unicode	Win/Mac/Linux	Konkordants, sõnaloend, märksõna-loend, concgrammid	Tasuline, aga demo ei ole väga piirav väikese korpuse jaoks
WordStat	JAH	Unicode	Win/Mac/Linux	Konkordants, statistika jm	Töötab moodulina SimStatiga või QDA Mineriga
PyPLN	JAH	Unicode	GNU/Linux	Konkordants, n-grammid, statistika jm	Inglise ja portugali keele jaoks

Testimiseks valitud programmide analüüsi tarbeks olen koostanud kriteeriumid, millel tuginedes uurin programmi installeerimise protsessi (1), korpuse laadimist programmi ja selle kasutamisviise (2), päringu koostamist ja töötlemist (3), funktsioone (4) ning tulemuste salvestamist (5). Jälgin, kuidas programm tuleb toime eesti tähestikuga, kuivõrd on võimalik tugineda juhendmaterjalil ja millised probleemid programmide kasutamisel tekivad.

Kriteeriumid programmide testimiseks:

(1) Programmi installeerimine: Kuidas programmi arvutisse installeerimine käib? Juhendmaterjal. Kodeering ja ühildumine eesti tähestikuga.

(2) Päringu koostamine ja töötlemine: Kas on võimalik kasutada metamärke ja regulaaravaldisi? Kas programm eristab suur- ja väiketähti? Kuidas seda muuta saab? Kas saab määrata, et otsitaks iseseisvat sõna (et tulemuste hulgas poleks sõnaosi) või vastupidi? Kas päringutulemuste hulka saab suurendada-vähendada? Millised võimalused on päringu koostamisel veel?

(3) Korpuse laadimine ja kasutamine: Kuidas käib korpuse programmi laadimine? Kas korpus võib olla ühes või mitmes tekstifailis? Milline võib olla korpuse formaat? Kas korpust on võimalik vahepeal vahetada? Kas märgendusi on võimalik peita?

(4) Funktsioonid: Millised funktsioonid on programmil? Analüüs iga funktsiooni kohta ning katsed. Kas ja milliseid üldstatistilisi andmeid on võimalik saada korpuse kohta?

(5) Tulemuste salvestamine: Kuidas saab tulemusi salvestada? Milline peab olema väljundi formaat?

2.3. Korpuse analüüsi programmide analüüsimine kui tarkvara testimine

Tarkvara testimine on protsess, mille eesmärgiks on leida, milline on tarkvara vastavus sellele esitatud nõuetele, kas ja kuidas tarkvara täidab oma eesmärki ning milliseid vigu sealt leida võib. Testimine on oluline juba tarkvara loomise algusetappides, sest leitavate vigade parandamine ei ole siis liiga keeruline ega kulukas. Tarkvara testimine aitab kaasa kvaliteetse ning kasutajat rahuldava toote valmimisele. (Graham jt 2008: 6, 13–14, 16)

Testimine on oluline osa tarkavaraarendusest ning et see oleks võimalikult tulemuslik, on loodud arenduse eri etappideks erinevad tarkvara testimise tehnikad. Nii saab leida igale etapile omaseid vigu. Kategooriaid on kaks: staatiline ja dünaamiline. (Graham jt 2008: 84) Staatilise testimise käigus uuritakse tarkvara manuaalselt, seda käivitamata. Selliseks testimise viisiks on näiteks läbivaatused. Seda tehnikat kasutades on võimalik leida tarkvaras esinevaid vigu, kuid lisaks anda ka üldist informatsiooni ja selgitusi projekti kohta. Sellised algstaadiumis tehtavad testimised aitavad vähendada hilisemat testimisele ja hooldamisele kuluvat aega. (Graham jt 2008: 58, 85)

Dünaamiline testimine jaguneb kolmeks: funktsionaalne, struktuurne ning kogemusepõhine testimine. Funktsionaalne testimine, mida kutsutakse ka musta kasti testimismeetodiks, jälgib seda, mida tarkvara teeb. Struktuurne testimisviis ehk valge kasti testimine analüüsib aga tarkvara sisesüsteemi ehk seda, kuidas programm töötab. Kogemusepõhise testimise puhul on määravaks see, millised on testivate inimeste teadmised, oskused ja taust. (Graham jt 2008: 85–86)

Funktsionaalse testimise üks alaliik on kasutusmallide testimine *(use case testing)*, mis on üheks selles töös kasutatavaks meetodiks. Selle testimismeetodi puhul kirjeldatakse programmi kasutamist ning seda, kuidas süsteem mingi ülesandega toime tuleb. Testimisel on võimalik avastada vead, mis reaalsetel kasutajatel programmi esmakordsel kasutamisel ette võivad tulla. (Graham jt 2008: 103–104) Sellega sarnane lähenemine, mida arvestan, on kasutatavuse testimine *(usability testing)*, mis aitab

tuvastada, kui arusaadav ja kui hästi kasutatav on programm. Oluline osa on programmi kasutajaliidesel, mille testimine tähendab olla kasutajarollis ning leida probleemseid kohti. Hea kasutajaliides järgib standardeid (näiteks Windowsi operatsioonisüsteemi omi) ja on ülesehituselt standardne teiste programmidega. On oluline, et liidest oleks võimalik kasutada intuitiivselt, eelnevatele kogemustele tuginedes: et kasutaja saaks ka dokumentatsiooni lugemata aru, kus mingi funktsioon asub. Lisaks peab kasutajaliides olema mugav ja paindlik ning sisaldama vajalikke osi. (Patton 2005: 169–173)

Kogemusepõhise testimise alla kuulub uuringtestimine *(exploratory testing)*, millel töös samuti põhinen. Uuringtestimise eesmärgiks on läbi kasutamiskogemuse programmi tundma õppida: kuidas seda kasutada, mida sellega teha saab ning millised on programmi negatiivsed-positiivsed küljed. (Graham jt 2008: 113)

Tarkvara testimisel tuleb lisaks tarkvara enda testimisele kontrollida üle ka muu sellega seonduv ning üheks selliseks on dokumentatsioon ja selle läbivaatamine. Korrektne dokumentatsioon muudab toote paremini kasutatavaks ja töökindlamaks, tekitamata kasutajale või tootjale lisakulusid. Dokumentatsiooni on kasulik testida viisil, nagu kasutab seda tavakasutaja: juhendmaterjali tuleks lugeda ja teha läbi seal toodud näited. Oluline on, et dokumentatsioon vastaks täpselt programmi funktsionaalsusele. Tuleks jälgida, kas see on kirjastiililt ja terminite kasutuselt sobiv selle lugejaskonnale ja katab kõik olulised teemad, kas informatsioon on tõene ning kirjavigadeta. (Patton 2005: 183, 187–189)

Erinevate kirjasüsteemide kasutamisel tuleb testida ka seda, kuidas programm võimaldab inglise tähestikku mittekuuluvate tähemärkidega töötada. Parimaks lahenduseks on kujunenud Unicode'i standard. Teised lahendused – nagu erinevad kooditabelid eri tähestike jaoks või DBCS *(Double-Byte Character Set)*, mis võimaldab kahebaidise süsteemiga kasutada üle 65000 erineva sümboli – ei ole kõige efektiivsemad, sest palju vigu tekib nende ühildamisel erinevaid tähestikke kasutavate süsteemidega. Unicode'i standardi omaduseks on see, et igal tähemärgil on oma kindel numbrikood. Nii on võimalik kasutada erinevaid tähestikke Unicode'i toetavates programmides. Testimine on oluline aga isegi siis, kui programm kirjade järgi justkui

toetab Unicode'i. Testimisel tuleks erinevatest tähemärkidest koosnevaid päringuid teha erinevates tekstiväljades, et näha, kas erimärgid töötavad samal moel nagu tavalised tähemärgid. (Patton 2005: 156–157)

Korpuse analüüsi programmide testimisel jälgin seega, mida ja kuidas programmid teha võimaldavad ning milliseid võimalikke probleeme testimine neis esile toob. Programmide kasutamisel toetun nende dokumentatsioonidele leidmaks, kas need sisaldavad kogu olulist infot ja aitavad kasutajat programmiga töötamisel. Päringu koostamisel ja funktsioonide kasutamisel jälgin, et programmid töötaksid kogu eesti tähestikuga.

3. Internetist korpuse koostamise programmide testimine

Oma korpuse loomiseks vajaliku tekstimaterjali saamiseks on erinevaid võimalusi. Üheks selliseks on tekstide kogumine internetilehekülgedelt. Lehekülgedelt on võimalik vajaminev tekst kopeerida ja kleepida tekstifaili, kuid see on ajakulukas ja ebamugav. Internetist tekstilise info kogumiseks on loodud programme, mis laadivad internetilehekülgedelt info ja salvestavad failina arvutisse, vähendades nii kasutaja käsitsi tehtavat tööd korpuse koostamisel. Töö raames valminud korpuse koostamisel olen testinud kolme programmi, milleks on BootCaT, SketchEngine'i versioon WebBootCat ning programmi TextSTAT internetist korpuse koostamise funktsioon.

3.1. BootCaT

BootCaT on tasuta allalaaditav programm, mis võimaldab kasutajal internetist märksõnade või internetilehekülgede kaudu oma korpust luua. Programm on mõeldud eelkõige spetsiaalkorpuste loomiseks. Näiteks lihtsustab BootCaT korpuse loomise protsessi juhtudel, kus on pidevalt tarvis uuendatud andmetega tekstimaterjali. (Baroni, Bernardini 2004)

Esmalt testisin programmis URL-loendi meetodit, kus tuleb soovitud lehekülgede internetiaadressid kopeerida ühte tekstifaili ning see fail programmi laadida. Esimesena proovisin koostada korpust Postimehe majandusuudiste leheküljelt pärit uudistekstidega. Valminud korpuse kontrollimisel selgus tulemusena, et programm oli üles leidnud kõik uudistekstid.

Proovides programmi samal viisil foorumitest pärinevate tekstidega, tulemus samaväärne ei olnud. Korpuse kontrollimisel, selle võrdlemisel foorumitega selgus, et

foorumitest ei kogu programm sageli kõiki postitusi. Näiteks oli foorumiteemasid, kust oli korpusesse lisatud vaid osa postitusi või mõnel juhul isegi vaid üks postitus, kuigi teemas oli kasutajate postitusi rohkem. Mõne foorumi postitusi on korpuses topelt: esimene kord tavapäraselt, teisel korral aga on tekst <*p*> ja </p> märgendite vahel. Kui neid foorumiteemasid aga vaadata, siis seal ei ole topelt postitusi ega ka tsiteeritud postitusi, mistõttu selline esitus korpuses põhjendatud ei ole.

Foorumite tekstidest saadav korpus ei ole parem ka märksõnade funktsiooni kasutades, mille puhul programm etteantud sõnade järgi ise internetist leheküljed välja selekteerib. Kuna märksõnade meetodi korral kasutab BootCaT otsingumootorit Bing, siis kehtivad turvalisuse kaalutlustel piirangud päringute tegemisele. Seetõttu tuleb funktsiooni kasutamiseks internetilehekülje Windows Azure Marketplace registreeruda kasutajaks, mis piirab päringute koostamise hulka ühes kuus 5000 päringuni.

Nii URL-ide kasutamisel kui ka märksõnade järgi otsides koostas programm foorumitekstidest korpuse, kus suur osa postitustest on jäänud mitme foorumi puhul esitamata. Seetõttu võib arvata, et BootCaT-i tekstide kogumise võimalused sõltuvad foorumist ja selle ülesehitusest.

Programmi BootCaT on üldjuhul lihtne kasutada. Ebamugavused tekivad, kui on tarvis korpust uuesti koostada või korraga mitut korpust luua. Iga kord, kui korpus on koostatud, tuleb programm sulgeda ning kui on tarvis kogu protsess uuesti läbi teha, siis taasavada (Selline vajadus võib tekkida, kui näiteks tekste ei kogutud soovitud suurusega korpuse jaoks piisavalt. Programm teavitab sellest, kui vastav funktsioon on sisse lülitatud). Kasutajal ei ole võimalik URL-meetodi puhul täiendatud internetilehekülgede faili eelmise asemele laadida, vaid peab ikka otsast alustama. Samuti ei saa valida, kas lehekülgedelt saadud tekstid lugeda ühte faili või mitmesse. Programm ei ole seega kuigi paindlik ning selline töötamise viis kujuneb üsna ajakulukaks, kui kasutajal tekib tarvidus muudatuste tegemiseks.

Programmi koostatavate korpuste ülesehitus on selline, kus iga uue teksti ees on kirjas *CURRENT URL + link,* mis pole märgenditega eraldatud. Lingi ja põhiteksti vahel ei ole tühikut ning uue lõigu alguses puudub lausete vahel samuti tühik.

Võib öelda, et BootCaT on lihtsalt kasutatav ja arusaadav programm, mis on sobiv tekstide kogumiseks lehekülgedelt, kus on olemas selgelt eralduv pikem tekst, näiteks uudisteportaalid. Foorumitest kogutud korpus ei anna oma puuduvate postitustega uurija jaoks selget ülevaadet materjalist, mistõttu selliste tekstide kogumiseks programm ei sobi.

3.2. WebBootCat

Et BootCaT ei andnud foorumitekstide puhul soovitavat tulemust, lisandus testimisse programmis SketchEngine sisalduv WebBootCat. SketchEngine on tasuline programm, kuid pakub tasuta kuuajalist prooviversiooni, mis on piisav aeg korpuse koostamiseks vajalike tekstide allalaadimiseks. WebBootCati saab kasutada brauseris, seega ei ole tarvis programmi arvutisse laadida ja vajalik on ainult kasutajana registreeruda. Sarnaselt BootCaT'ile on ka WebBootCati eesmärgiks pakkuda võimalust koostada kiirelt korpust eri uurimisvaldkondades. (Baroni jt 2006)

Testisin WebBootCatiga nii uudis- kui foorumitekstide korjamist soovitud lehekülgede internetiaadresside lisamisel programmi (WebBootCatis on võimalik URL-aadresse lisada juurde või kustutada igal ajal). Selgub, et programm korjab tekste edukalt nii uudiste lehekülgedelt kui ka foorumitest. Kõik Postimehe uudised on täispikkuses korpuses olemas ja tekst selgelt liigendatud ning ka enamik foorumite postitusi on korpuses esindatud.

Foorumipostitustega tuli korpuses esile probleem, kus üht postitust oli mitmekordselt. See ilmselt oleneb foorumist, sest kontrollides internetilehekülgelt, polnud ükski postitus mitmekordne ning puudusid ka tsiteeritud postitused (sama probleem oli ka BootCaT'iga). Mõnes kohas on foorumipostitus puudu, kuid paistab, et see on seotud postituse pikkusega. Programm ei olnud korpusesse lisanud näiteks selliseid laused nagu *Merisead hammustavad samuti! Ainult neil on suuremad hambad; loomake otsib sooja ju*. Vaadeldes teisi postitusi samas foorumiteemas, olid need nähtavalt pikemad

kui eelkirjutatud korpusesse lisamata jäänud laused. Seetõttu võib arvata, et programm ei arvestanud pikkade postituste kõrval selliseid lühikesi lauseid tekstina ega lugenud neid seepärast korpusesse.

Mõningal määral võib korpusest leida ebavajalikku teksti. Ühe foorumi puhul on korpuses märkused kujul *Deprecated: Assigning the return value of new by reference is deprecated in /home/np7910/domains/koer.ee/public_html/components/com_kunena/kunena.php on line 691*, kus iga märkuse korral reanumbrid erinevad. Tegu on ilmselt veateatega, sest real 691 on kodulehel postitus, milles puudub tekstiline sisu. Lisaks postitustele on mõnes kohas ka foorumikasutaja staatus, mis tavaliselt foorumis postituse all näha on, näiteks mõni ingliskeelne tsitaat. Delfi foorumist kogutud postituste juures võib näha ka informatsiooni postituse kirjutamise kohta (seda esines ka BootCaT'i koostatud korpuses), näiteks *Selleks, et lisada oma postitusele pilt, video või pildialbum, kopeeri postituse väljale pildi, video või albumi aadress*.

Loodud korpuses on tekstilõikude algused ja lõpud tähistatud märgendite <*p*> ja </*p*> abil. Kolmnurksulgude vahel on internetilehekülje aadress. Võrreldes BootCaT'i foorumitest kogutud korpusega on WebBootCat koostanud korpuse, kus enamik foorumite postitustest on korpuses olemas, kuigi nende seas võib esineda ka tekstimaterjali, mida otseselt tarvis ei ole. Seega sobib WebBootCat internetist tekstide kogumiseks, sest suudab koguda tekste erinevatest allikatest.

3.3. TextSTAT

Kasutajal on mugav kasutada üht programmi, mis võimaldab korpuse koostamise kõrval seda kohe ka analüüsida, seetõttu proovisin internetist kogumise funktsiooni programmiga TextSTAT, millel on ka konkordantsi ja sõnaloendi koostamise võimalused. Programmi on vaja sisestada lehekülgede internetiaadressid ning seejärel on korpust võimalik kohe kasutada. Tulemust ei saa siiski nimetada hästi koostatud korpuseks, sest kõrvalist tekstimaterjali on palju nii foorumite kui ka uudiste

lehekülgedelt. Nii on programm korpusesse lisanud teiste uudiste pealkirjad ning foorumitest kogu kasutajainfo, näiteks *Postitamiseks pead olema sisse logitud. üles #7383 kadivalder Karma: 0 Koeraja Postitusi: 1 Offline Profiil 6 kuud, 2 nädalat tagasi*. Et loodud korpust programmiväliselt näha, tuleb salvestamisel lisada txt-laiend.

Proovitud TextSTAT-i funktsioon on sobilikum näiteks juhul, kui on tarvis saada kiire ülevaade mõne internetilehe keelekasutusest, sest programmi kasutamine oli kerge ja sujuv. Põhjaliku uurimistöö tarbeks kasutamiseks sisaldab loodav korpus liialt kõrvalist tekstimaterjali, mis segab töö jaoks olulise informatsiooni uurimist.

Internetist tekstide kogumise funktsioon WebGetter on olemas ka programmil WordSmith Tool, kuid siinse töö kirjutamise hetkel oli see arendamisel ega töötanud.

4. Korpuse analüüsi programmide testimine

Korpuse analüüsi programmide testimisel järgin peatükis 2.2 kirjeldatud kriteeriume. Igas kategoorias kirjeldan programmide Simple Concordance Program (SCP), AntConc, MonoconcEsy, Multilingual Corpus Toolkit (MLCT) ja TextSTAT testimise tulemusi. Kirjeldamisel lähtun sellest, et see oleks kasutajale justkui lisa juhendmaterjal programmiga töötamisel.

Iga kategooria võrdlustulemuste järel toon välja, millised programmid töötavad selles protsessis või mingi funktsiooni kasutamisel kõige paremini, arvestades ka kasutajakogemust ning programmi enda juhendmaterjali järgi töötamise võimalust. Analüüsi tulemusena selguvad programm(id), mis on kõige sobivam(ad) oma korpuse analüüsimiseks.

4.1. Programmide installeerimine

Installeerimise protsess on kõigi viie programmi puhul suhteliselt kiire ja igale tavakasutajale võimetekohane. Simple Concordance Programi koduleheküljel valisin sellise Windowsil töötava 4.0.9 versiooni, mille arvutisse laadimine käib installeerimisaknaga. MonoconcEsy, Multilingual Corpus Toolkit ja TextSTAT 2.9 kasutavad installeerimiseks zip-faili, mille lahti pakkimisel failil klikates tööriist kohe tööle hakkab. AntConci installeerimiseks tuleb alla laadida fail, mida käivitades hakkab programm tööle ja lisavalikuid teha pole tarvis.

Programmide SCP, AntConc ja MLCT juhendmaterjalid leiab *Help* nupu alt. MonoconcEsy juhend asub samas kaustas koos programmifailiga. TextSTAT-i dokumentatsioonile on koduleheküljel küll viited olemas, kuid ükski ei suuna enam

funktsioneerivale leheküljele. Juhendi leiab, kui sisestada Google'i otsingumootorisse *TextSTAT guide*, nii on võimalik alla laadida TextSTAT-i versiooni 2.7 juhend.

Eesti keele tähestikuga töötamisel tuleb kontrollida, kas programm toetab Unicode'i kodeeringut. Kui MLCT ja TextSTAT töötavad nii Unicode'i UTF-8 kui ka UTF-16 kodeeringuga, siis AntConc töötab vaid UTF-8 failidega ja MonoconcEsy eeldab UTF-16 kodeeringut. SCP-s eesti tähestiku jaoks sobivat kooditabelit valida ei saa, kuid vajalikud tähed saab ise tähestikku lisada (täpsemalt peatükis „Korpuse laadimine ja kasutamine").

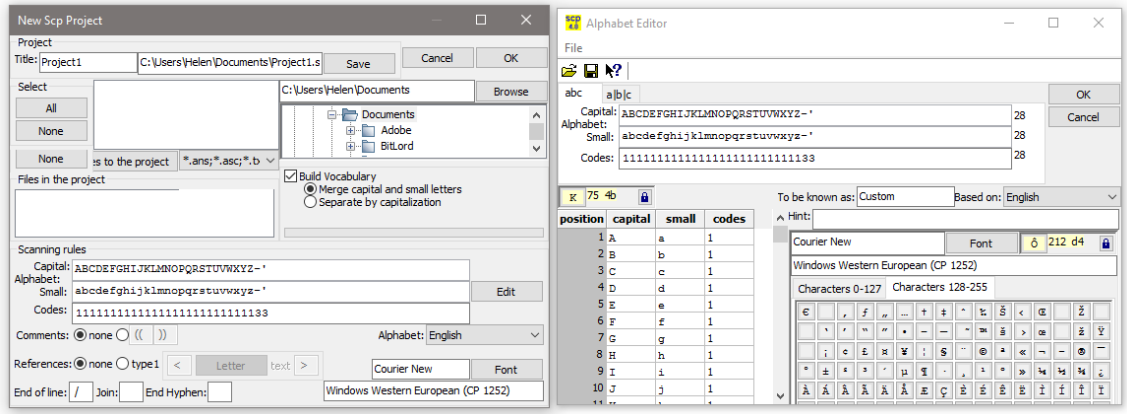
Kõigi programmide installeerimine on lihtne, kõige kiirem ja mugavam on AntConci paigaldamine, mida on võimalik käivitada ühe allalaetud faili abil. Juhendid on olemas, kuid TextSTAT-i jaoks tuleb see internetist eraldi otsida. Unicode'i kodeeringu kasutamise võimalus on olemas kõigil peale Simple Concordance Programi, millel on oma süsteem eri tähestikega töötamiseks.

4.2. Korpuse laadimine ja kasutamine

Oma korpuse laadimise ja kasutamise võimaluste poolest jagunevad programmid kaheks: SCP-s ja TextSTAT-is tuleb programmiga töötamiseks luua kõiki vajalikke faile sisaldav projekt, ülejäänud kolmes programmis tuleb failid programmi avamisel iga kord uuesti laadida. Kui korpusega on tarvis pidevalt töötada, on kindlate sätete ja failidega projekti programmi laadimine mugavam, kui iga kord korpust laadida ning vajalikke seadeid muuta.

Kõik programmid toetavad .txt-laiendiga faile, mis on kindlaim viis oma korpuse salvestamiseks. AntConc töötabki vaid txt-failidega, kuid ülejäänud programmid toetavad ka muid formaate (vt lisa 1). Kui korpusfailides on osa tekstist piiritletud mingite kindlate märkidega (nt < ja >), siis on programmides SCP, MonoconcEsy ja AntConc võimalik märke koos nende vahele jääva tekstiga peita. MLCT-s ja TextSTAT-is seda teha ei saa.

Korpuse laadimist on programmide dokumentatsioonis selgelt kirjeldatud ja juhiseid järgides on see protsess lihtne. Teistest keerulisem võib kasutajale olla projekti koostamine programmis SCP, sest lisaks muudele sätetele tuleb seal ise täiendada ka tähestikku, lisades täpitähed ja *š*, *ž*, et oleks võimalik eestikeelsete tekstidega töötada (vt joonis 2). SCP juhend on aga väga informatiivne: esitatud on protsessi kirjeldav näide ning lisaks on sätteid selgitatud, näiteks *End of line* ehk *realõpp* tähendus. MLCT-s tuleb aga korpus uuesti valida iga kord, kui näiteks n-gramme või kollokatsioone koostada.



Joonis 2. **Projekti koostamine programmis SCP.**

Kui korpus on programmi laaditud, võib kasutamise käigus tekkida vajadus seda muuta: kas mingi tekstifail eemaldada või juurde lisada. SCP projekti muuta ei ole võimalik, ka mitte määratud sätteid, mistõttu tuleb sellisel juhul luua uus projekt. MLCT-s tuleb selleks uuesti kõik failid korraga programmi laadida. Programmides MonoconcEsy, AntConc ning TextSTAT saab faile eemaldada või lisada igal hetkel.

Kõige kergem on programmide MonoconcEsy, AntConc ning TextSTAT korpuse laadimise protsess. Neist mugavaim on TextSTAT, sest esimesel kasutuskorral luuakse projekt, mis sisaldab valitud korpusfaile ka programmi uuesti avades ning faile on võimalik jooksvalt eemaldada või lisada. Miinuseks on märgenduse peitmise võimaluse puudumine.

4.3. Päringu koostamine ja töötlemine

Päringu koostamisel on oluline valikuvõimalus, sest nii saab kasutaja teha oma vajadustest lähtuvalt täpsema otsingu. Näiteks filtreerib tulemusi suur- ja väiketähtede eristamise võimalus. Testitud programmidest teevad seda kõik, kuid MLCT-s ei saa seda muuta. See tähendab, et kui otsingus on vaja eristada nii suur- kui ka väiketähega sõnu, siis ainus võimalus selleks on kasutada regulaaravaldist.

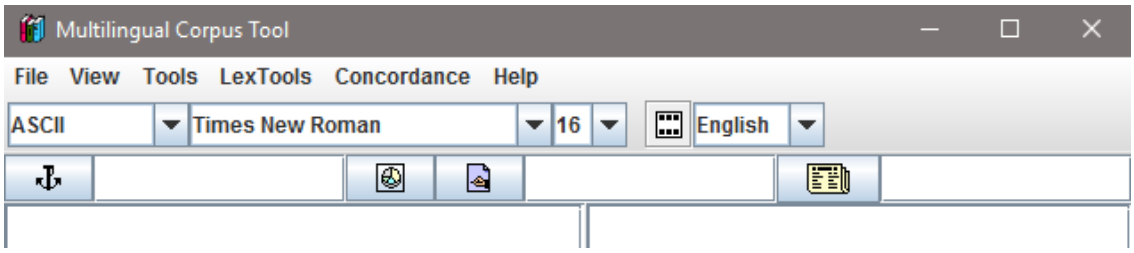
See olukord näitab, et regulaaravaldised teevad programmi kasutamise paindlikumaks ja võimaldavad teha täpsemaid päringuid, seetõttu on nende kasutamisvõimalus programmi suureks plussiks. Viiest testitud programmist ei toeta regulaaravaldisi vaid SCP.

Programmide kasutamisel tuleb arvestada, et need ei pruugi samu metamärke toetada. Programmis MonoconcEsy saab %-märgiga asendada otsingusõnas ühe tähemärgi, saades nii tulemuseks sellised sõnad/read, kus märgi asukohas on üks täht või pole ühtegi. Näiteks otsinguga *ja%* saab vasteteks nii *jah* kui ka *ja*. Teised programmid sellist regulaaravaldise süntaksit ei toeta. MonoconcEsy juhendis on hea ülevaade võimalikest programmis kasutatavatest märkidest ja näiteid on esitatud ka pikemate regulaaravaldiste kasutamisest. Teiste programmide juhendites regulaaravaldiste kohta informatsiooni ei ole: AntConcis on viidatud võimalusele otsida lisainformatsiooni internetist, MLCT juhendis on mainitud vaid nende kasutamisvõimalust ning TextSTAT-is on küll olemas regulaaravaldiste abile viitav nupp, kuid selle kaudu suunatakse mittetöötavale internetilehele.

Päringu koostamisel on oluline ka võimalus otsida terviklikke sõnu ja välistada päringutulemustes sõnaosi. AntConcis ja TextSTAT-is on see võimalus olemas (vastavalt *Words* ja *whole words only*). SCP võimalused on suuremad. Otsida on võimalik eesliidet, järelliidet, sõna või hoopis nii, et otsingusõna paiknevus päringu-tulemustes pole oluline (tulemustes on märksõna siis ka sõnaosa). MonoconcEsys on päringuvastetes ainult terviksõnad ja MLCT-s lisaks terviksõnadele ka sõnaosad. Vaikimisi seadeid muuta ei ole võimalik. MonoconcEsys tuleb sõnaosana esinevate

otsingutulemuste saamiseks kasutada näiteks mõnd metamärki. MLCT-s saab ainult terviksõnad tulemustes siis, kui lisada tühikud märksõna ette ja järele. Peab arvesse võtma, et nii ei väljastata tulemusi, kus tühik ei eralda sõnast kirjavahemärki. Regulaaravaldisega on see aga võimalik.

MonoconcEsy on ainus programm, kus on võimalik kuvatavat päringutulemuste arvu suurendada või vähendada. Programm MLCT erineb teistest selle poolest, et päringu tegemiseks on kolm akent: konkordantsi päringu ja regulaaravaldise sisestamise aken ning lahter, mis on mõeldud regulaaravaldiste asendamiseks või mustrite muutmiseks. Kolme eri akna kasutamine ei ole eriti kasutajasõbralik (vt joonis 2).



Joonis 2. **Programmi MLCT kolm päringuakent.**

Päringu koostamise kohta puuduvad juhendites põhjalikud suunised. SCP-s ja MLCT-s pole selle kohta üldse informatsiooni, kuigi viimase programmiga on just päringu koostamine keerulisem. AntConci ja TextSTAT-i juhendmaterjalides on iga funktsiooni juures esitatud päringu moodustamise viisid. MonoconcEsy juhendis on kirjeldatud vastava protsessi sätteid ning olemas on info selle kohta, mida programm arvestab sõnana ja kuidas seda muuta saab.

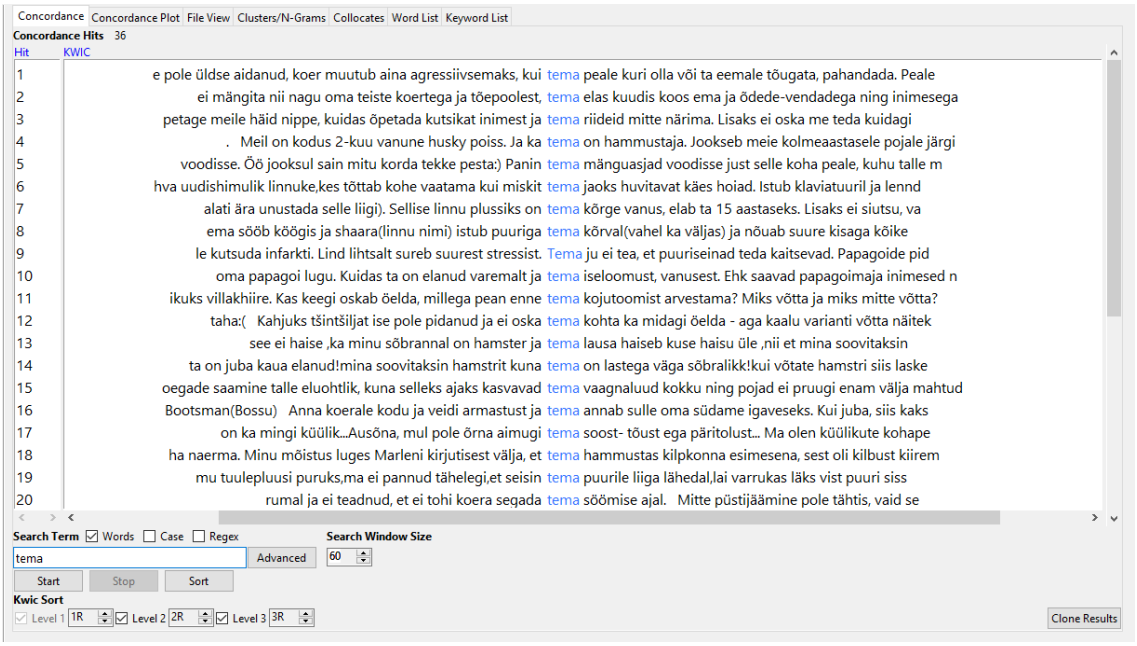
Kasutaja jaoks, kes pole regulaaravaldistega palju kokku puutunud, on oluline võimalus koostada päringuid neid kasutamata. Päringu koostamine on lihtsasti mõistetav ja põhjalikult dokumenteeritud programmis MonoconcEsy. AntConcil ning TextSTAT-il on kasutajasõbralik päringu koostamise liides ning lahtritel ja nuppudel on informatiivsed nimetused.

4.4. Programmide funktsioonid

Programmid pakuvad korpuse uurimiseks ja tekstile iseloomulike tunnuste tuvastamiseks erinevaid võimalusi. Seepärast tuleb enne programmi valimist olla kindel, et sellega on võimalik soovitavat meetodit kasutada. Korpuse analüüsi programme kasutatakse peamiselt konkordantsi koostamiseks, kuid enamikus programmides on võimalik teha ka sõnaloendit. Töös testitud programmides saab veel koostada kollokatsiooni, n-gramme ja märksõnaloendit. Programmid TextSTAT ja MLCT võimaldavad ka korpuse koostamist internetist.

4.4.1. Konkordants

Kõigis testitud programmides on võimalik koostada konkordantsi[4] (vt joonis 3), kuid saadav tulemus erineb programmiti.



Joonis 3. **Konkordants programmis AntConc.**

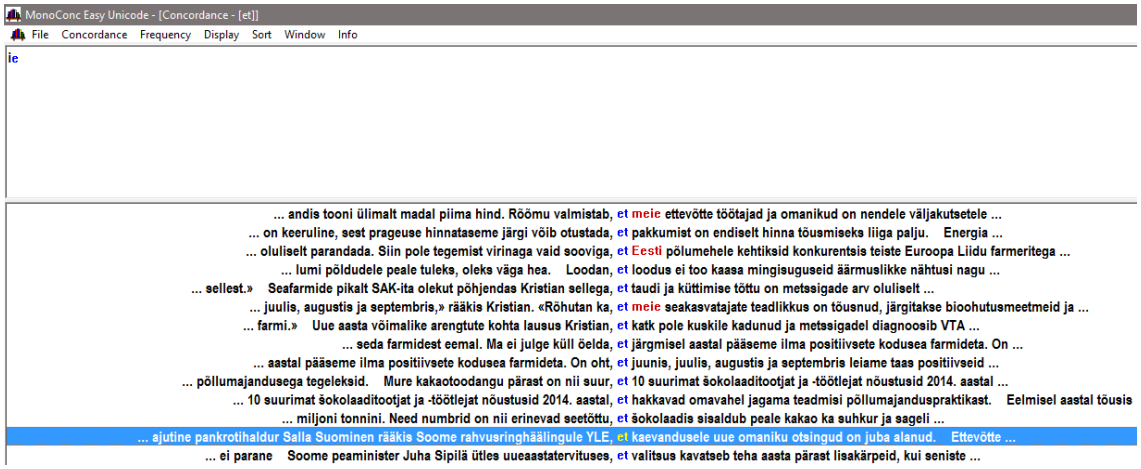
Konkordantside leidmiseks on tarvis koostada päring. MonoconcEsy, AntConci ja TextSTAT-i juhendites on konkordantsi koostamine esitatud näidete ja etappidena. SCP

[4] Konkordants on sõnavorm koos kontekstiga (lk 6).

juhendis on väga üldiselt kirjutatud konteksti määramise ja mingi sõnahulga kasutamise võimalustest, sorteerimisest ning otsingu koostamisest. Samas pole konkordantsi juures selgitatud valikuid *Keys*, *anyKey*, *Phrase*, *Series*, *allKeys*, mille tähendus ja kasutamisviis jäävad arusaamatuks. MLCT juhendis on kirjutatud konkordantsi sorteerimisest ning menüüst, kust korpust laadida, kuid koostamise kohta info puudub.

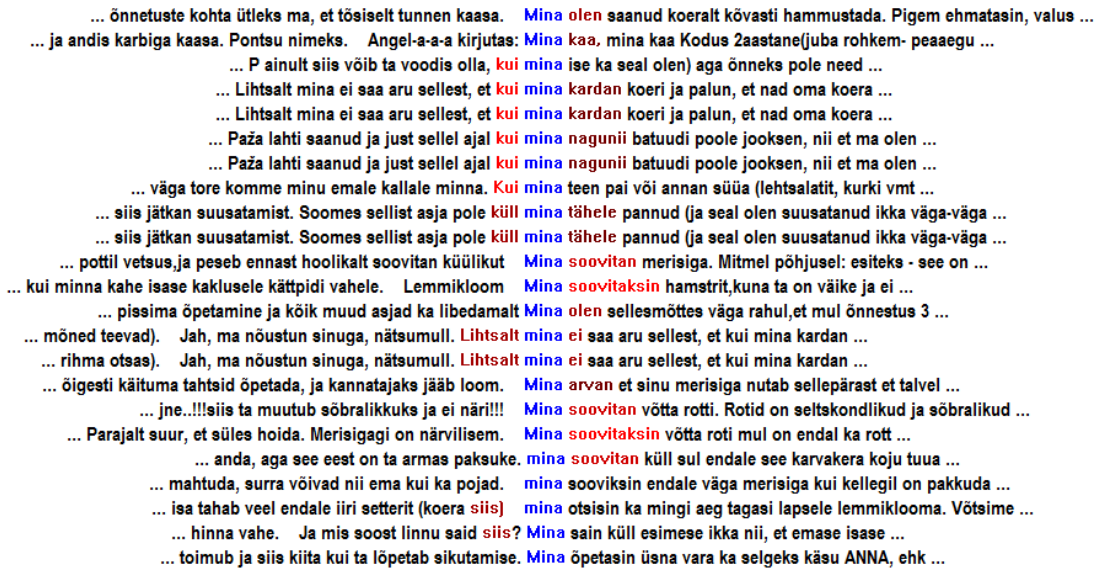
Konkordantsi oluline osa on kontekst märksõna ümber. Kõikides programmides on võimalik konteksti hulka muuta. MLCT-s on konteksti tähemärkide arv piiramatu, AntConcis ja MonoconcEsys on piiriks 1000 tähemärki. Viimases on võimalik konteksti hulka määrata veel ka sõnade, lausete ja ridade alusel. TextSTAT-is saab eraldi valida, kui palju konteksti võimalikust 10–100 tähemärgist esitada vasakul, kui palju paremal pool. SCP-s on konkordantsi saamiseks tarvis vajutada kas nupul *Kwic* või *Line*, millest viimane näitab sõna ümber rohkem konteksti.

MonoconcEsy, AntConc ning TextSTAT võimaldavad näha, kus mingi märksõna korpuses asub: ühel real paiknevale märksõnale vajutades saab näha seda osa korpusest. AntConcis ja TextSTAT-is avaneb see teises aknas, nii et vajadusel on võimalik tagasi pöörduda konkordantsi tulemuste juurde, MonoconcEsyl on see koht tulemuste kohal olevas aknas. Lisaks on AntConcis võimalik näha ka üldist jaotust, kuidas päringu-tulemused kogu faili ulatuses paiknevad. MonoconcEsy küll võimaldab juhendi järgi näha valitud kohta failis, kuid nagu joonisel 4 on näha, kasutamisel see ei tööta: programm esitab selles aknas vaid paar suvalist tähekombinatsiooni. Programmidel SCP ja MLCT selline võimalus puudub.



Joonis 4. **MonoconcEsy esitab faili vaates (vt ülemine aken) lause asemel suvalised tähekombinatsioonid.**

Konkordantsi sorteerimisel on võimalik näha sarnasel keelekasutusel tekkivaid mustreid. Kõige parem sorteerimisviis on programmidel MonoconcEsy ja AntConc: sorteerida saab mitmetasandiliselt ning värvidega on mustrid hästi nähtavad. MonoconcEsys saab sorteerida kaks sõna vasakule-paremale, AntConci piiriks on 20 sõna. MonoconcEsys konkordantsiridade sorteerimisel sõnu värviliseks ei muudeta, kuid kui sisse on lülitatud *Highlight Collocates*, siis on erinevad mustrid automaatselt esile toodud (vt joonis 5). AntConcis saab valida sõnade kõrval ka tähemärkide alusel sorteerimist, lisaks on võimalik eraldi grupeerida esisuurtähega sõnad. MLCT-s puudub mitmetasandiline sorteerimine, mistõttu on korraga võimalik kasutada ainult ühte sorteerimiskriteeriumit, näiteks paremalt esimese sõna alusel. SCP ja TextSTAT sorteerivad ridu tähemärkide alusel: võimalik on sorteerida nii, nagu programm read esitab või märksõna vasaku-parema poole alusel. See tähendab, et programm sorteerib tähestiku järgi kas paremalt esimese sõna esimese tähe järgi või märksõnast vasakule jääva sõna viimase tähe järgi.



Joonis 5. **MonoconcEsy automaatselt esile toodavad mustrid nupuga *Highlight Collocates*.**

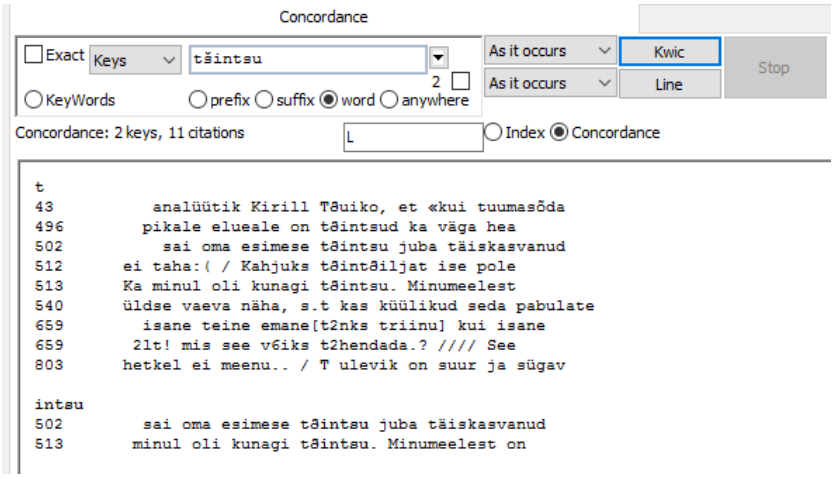
Kontrollimaks programmide konkordantsi tulemusi ja nende erinemist programmiti, testisin konkordantse märksõnadega *kõik*, *lõi*, *mina*, *sellepärast*, *sellepärast et*, *tšintsu*, *šokolaadi*, *kui*. Katsesse valitud sõnade abil on võimalik näha, kuidas programmid töötavad tähemärkidega *š* ja *ž* ning täpitähtedega, milliselt arvestatakse fraase ning kui sarnased on tulemused selliste sagedaste sõnadega nagu *mina* ja *kui*.

Sõnaga *lõi* probleeme polnud ning kõik viis vastet tulid välja igas programmis. Sõna *kõik* otsingul tuli aga programmidel vasteid 42 või 46. Uurimisel selgus, et programmidel MLCT ja MonoconcEsy, kus tuli 42 tulemust, jäävad arvestamata sellised sõnad, mille küljes on erinevad kirjavahemärgid. MonoconcEsys aitab eraldajate *(delimiters)* lisamine vastavasse loendisse, pärast mida tuleb vasteid 46 nagu teistel programmidelgi[5]. MLCT-s eraldajaid ei saa määrata, kuid regulaaravaldise abiga on võimalik tuvastada ka kirjavahemärkide küljes olevad sõnad. Sel põhjusel on MLCT-s tulemused erinevad ka sõnadega *mina* ja *sellepärast*.

Fraasi *sellepärast et* otsimist katsetasin, et näha, kas programmid arvestavad vaid seda varianti või ka fraasi *sellepärast, et*. Fraasi otsimiseks tuleb SCP-s kasutada valikut *Phrase* ning nii väljastab programm kaks vastet, komaga ja komata variandid.

[5] Järgnevate katsete puhul on MonoconcEsys tehtud päringud koos eraldajatega.

TextSTAT-is tuleb nuppu *Query Editor* kasutades tulemeid kaks, samamoodi ka AntConcil ja MonoconcEsyl. MLCT ei ole arvestanud komaga fraasi *sellepärast, et* põhjusel, et sõnal *sellepärast* on punkt küljes. Sõnade *tšintsu* ja *šokolaadi* otsingutulemused on kõikidel programmidel samad, välja arvatud SCP-l. Ilmneb, et programm ei tööta *š-* ega ka *ž*-tähega. Näiteks sõna *šokolaadi* otsimisel on näha, et programm on järginud mustrit *okolaadi* ning joonisel 6 võib näha, et *tšintsu* puhul on otsinud eraldi *t* ja *intsu*.



Joonis 6. **Programm SCP ei tööta *š-* ega *ž*-tähega, näide sõnaga *tšintsu*.**

Sõna *kui* otsingutulemused olid programmidel üsna erinevad: AntConc ja SCP andsid 344 vastet, MLCT 328, TextSTAT 346 ning MonoconcEsy 343 vastet. Et teha kindlaks korrektne vastus, lugesin kokku korpusest kõik *kui*-d ning selgus, et selliseid *kui*-sid, millel ühtki märki küljes ei ole, on kokku 329 (olenemata suur- või väiketähest). Sõna *kui* esinemisi nii eraldi kui ka koos kirjavahemärkidega oli kokku 344. Seega arvestavad AntConc ning SCP kõiki võimalikke variante, MLCT on aga tuvastanud kirjavahe-märkideta *kui*-d.

Kuidas aga TextSTAT sai kaks vastet õigest tulemusest rohkem ning MonoconcEsy ühe vähem? Kuna TextSTAT-is ei ole võimalik märgendeid peita, siis on programm arvestanud ka sellised *kui*-d, mis asuvad URL-is märgendite vahel. MonoconcEsy puuduv lause oli aga selline, kus *kui* järel ei olnud mitte tavapärane tühik, vaid mingi muu mittetäheline sümbol ning kuna sellist sümbolit ei olnud eraldajana lisatud, siis

programm seda otsitud märksõnana ei esitanud. Sama lause puudus ka programmil MLCT.

Võib öelda, et konkordantsi tulemused on programmidel sarnased, kuid erinevad teatud määral programmidel, mis ei tõsta automaatselt kirjavahemärke sõnadest eraldi (MonoconcEsy ja MLCT). Katse tõi esile ka selle, kuidas programm ei pruugi eesti tähestikuga töötada, isegi kui kirjelduses on öeldud, et sobib töötamiseks erinevate keeltega – SCP-s ei saa kasutada eesti tähestikku kuuluvaid tähti *š* ja *ž*.

Kasutajana oli konkordantsi koostamine ebamugavam programmides SCP ja MLCT. SCP-s oli keeruline aru saada, kuidas õige päringutulemuse saamiseks eri valikuid omavahel ühildada, sest mõned neist polnud juhendis lahti seletatud. MLCT kasutaja-liides on raskesti mõistetav ning kuigi konkordantsi on programmis ehk lihtsam koostada kui mõnd teist funktsiooni kasutada, on programmis vähe sätete muutmise võimalusi. TextSTAT-is oli konkordantsi kasutamine küll lihtne, kuid programmis pole võimalik märgendusi peita, sorteerimisvõimalusi on võrreldes teiste programmidega vähem ning märksõna ja mustrite eristamiseks ei kasutata värve, mis teeb need silmale raskemini haaratavaks. MonoconcEsys on konkordantsi lihtne koostada, päringu tegemisel on mitmeid võimalusi ning otsingulahtri juures on abistavad näited. Hea on ka see, et programm näitab konkordantsiridu sorteerimatagi tekkivaid mustreid. Ridade ees aga puuduvad numbrid. AntConcis on konkordantsi koostamine lihtne ning on mugav, et muudetavad sätted on kõik ühes kohas. Seega paistavad programmide hulgast konkordantsi kasutamisel välja MonoconcEsy ja AntConc, kuid kuna MonoconcEsy ei eralda kirjavahemärke automaatselt sõnadest, siis võib AntConci pidada selle funktsiooni kasutamisel kõige paremaks ja mugavamaks.

4.4.2. Sõnaloend

Kõik testitud programmid võimaldavad koostada sõnaloendit[6], kuid funktsiooni nimetus võib programmiti erineda: on kasutatud kas sõnaloendi *(wordlist)* või sagedusloendi *(frequency list)* mõistet, mis programmides annavad tulemuseks samasugused loendid. Programmil MonoconcEsy on see funktsioon kirjas kui *Corpus Frequency data*.

[6] Sõnaloend näitab, mis sõnad millise sagedusega korpuses esinevad (lk 6).

MLCT-s on sõnaloend leitav n-grammi funktsiooni alt. Töös kasutan ühtluse huvides sõnaloendi mõistet.

MonoconcEsy juhend kirjeldab kõige täpsemalt sõnaloendi koostamise kasutamis-võimalusi, lisaks on juures pildid ning näited. AntConci dokumentatsioonis on esitatud koostamisetapid ja TextSTAT-i juhendis on pildid, mille juures on punktidena välja toodud erinevad valikud koos tähendusega. MLCT juhendmaterjalis on selgitatud vaid menüüde nimetusi ehk mille jaoks mingi menüü mõeldud on. SCP juhendis on kirjas, kus sõnaloendit koostada ja millised võimalused sealjuures on. See on piisav mõistmaks, kuidas loendit luua. Samas on jäetud juhendisse kirjutamata, et sõnaloendi tegemiseks on tarvis koostada sõnastik *(Build Vocabulary),* mille jaoks on nupp algses projekti koostamise aknas. Kui seda kohe projekti loomisel pole tehtud, tuleb teha uus projekt, mis on selles programmis üsna tülikas protsess.

Sõnaloendi moodustamisel on programmides SCP ja AntConc lisavalikuna võimalik valida välja sõnad, mille põhjal sõnaloend koostada, või eemaldada mõned, mida loendisse arvestada ei soovi. MonoconcEsys on võimalik vaid viimane variant. AntConcis on mugav võimalus kirjutada otsinguribale sõna ning siis nupu *Search Only* abil leida selle asukoht sõnaloendis.

Sõnaloendit saab programmides sorteerida sageduse järgi või tähestikuliselt. AntConcis ja TextSTAT-is saab sorteerida ka tähestikuliselt sõna lõpu järgi, SCP-s aga hoopis sõna pikkuse põhjal.

Testisin, kuidas programmid sõnaloendit moodustavad: vaatlesin sõnaloendite esimest kümmet tulemust, sorteerides sageduse alusel kahanevalt (vt tabel 2). Tulemus on ühesugune programmidel SCP ja AntConc. MonoconcEsyl on erinevused sõnadega *on, kui* ja *ta* (sõnade järjestus on siiski sama), kus nende kolme sõna esinemise hulk on ühe võrra väiksem ehk programm on jätnud arvestamata mõned laused. Kontrollimisel selgus, et sõna *on* vastetes on jäänud arvestamata lause, kus *on* paikneb sellises lausekontekstis nagu *Vene rahandusministeerium (...) on alustanud*. Sõna *kui* hulk erines lause tõttu, mis tekitas erinevaid tulemusi ka konkordantsis: õige tühiku asemel oli nähtamatu sümbol, mistõttu MonoconcEsy ei arvestanud *kui*-d iseseisvaks sõnaks.

Sõna *ta* arvestuses puudus lause, kus oli sõna *võta* kirjutatud kujul *v6ta*. Kuna ma MonoconcEsy eraldajate hulka numbreid ei lisanud, siis programm seda sõna numbri kohalt ei poolitanud. Ilmnes, et programmidel AntConc ja SCP on vaikimisi sõnade eraldajate hulgas ka numbrid.

Tabelist 2 on näha, et ülejäänud kahe programmi tulemused on väga erinevad. MLCT loendis on sõnade esinemise hulk väiksem, mis tuleb sellest, et programmis ei saa eraldajaid defineerida. Nii jäävad loendist välja sõnad, mille küljes on erinevad kirjavahemärgid või mittetähelised sümbolid. TextSTAT-i sõnaloendis on esimesel kohal *p*, sest programmis ei ole võimalik märgendeid peita ning *p* on osa lõigu algust ja lõppu tähistavast html-märgendist. Ka ülejäänud TextSTAT-i tulemused on üsna erinevad teistest programmidest, kuid põhjuseid pole võimalik leida: kui vajutada loendis näiteks sõna *ja* peal, et näha selle hulka arvestatud lauseid, avaneb konkordantsi aken. Seal ei näita enam programm mitte sõnaloendisse arvestatud lauseid, vaid konkordantsi tulemusi, need aga erinevad, sest leitakse ilmselt teistel põhimõtetel (sõnaloendis oli nt sõnaga *ja* tulemusi 792, konkordants annab aga 809 vastet). Eraldi sõnaloendis arvestatud lauseid näha pole võimalik.

Tabel 2. **Sõnaloendi katse.**

Programmid	**Sõna - esinemise hulk**									
SCP	ja - 805	on - 754	ei - 406	et - 382	kui - 344	ka - 310	ta - 245	siis - 241	aga - 202	see - 176
AntConc	ja - 805	on - 754	ei - 406	et - 382	kui - 344	ka - 310	ta - 245	siis - 241	aga - 202	see - 176
MonoconcEsy	ja - 805	on - 753	ei - 406	et - 382	kui - 343	ka - 310	ta - 244	siis - 241	aga - 202	see - 176
MLCT	ja - 739	on - 719	ei - 393	et - 360	ka - 276	kui - 274	ta - 210	siis - 208	aga - 152	oma - 143
TextSTAT	p - 1623	ja - 792	on - 750	ei - 404	et - 369	kui - 335	ka - 306	ta - 240	siis - 238	aga - 197

Lisaks sõnasagedusele on mõne programmiga võimalik leida muidki sagedusi. SCP-s saab näiteks leida tähesagedust ja erinevate sümbolite esinemise sagedust korpuses. MLCT-s saab koostada lause pikkuse sagedusloendit, kus on esitatud vastava

lausepikkusega lausete arv korpuses (nt et kahesõnalisi lauseid on 38). TextSTAT-is on võimalik sisestada märksõna või regulaaravaldis ja saada selle märksõna eri vormid listina. Programmis saab sõnaloendis suvalise sõna peal klõpsates näha selle sõna protsentuaalset osakaalu kogu korpusest.

Sõnaloendi moodustamisel töötavad arvestatavalt programmid SCP, AntConc ning MonoconcEsy. Et MonoconcEsys tuleb lisada vajalikud eraldajad, on ülejäänud kahte programmi mugavam kasutada. AntConc ega SCP ei sobi aga netikeele analüüsiks, milles on tavaks kasutada numbreid täpitähtede asemel. Tuleb ka arvestada, et SCP ei tööta *š*- ja *ž*-tähega, mis võib mõjutada analüüsitulemusi. Seetõttu võib kindlamalt kasutada programmi AntConc, mis läbis edukalt sõnaloendi moodustamise katse, mille juhend on abistav ning mida on mugav kasutada, sest vajalikud nupud on koondatud ühte kohta.

4.4.3. Kollokatsioon

Kollokatsiooni[7] koostamine on võimalik MonoconcEsys, AntConcis ja MLCT-s. MonoconcEsy ja AntConci funktsioon töötab nii, et otsingusse tuleb sisestada märksõna, mille kollokatsioonid leitakse. MonoconcEsys on olemas ka nupp *Highlight Collocates,* mis sisselülitatuna näitab juba konkordantsi koostamisel ära sagedasemad kollokatsioonid. MLCT-s mingi kindla sõna kollokatsioone otsida ei saa, vaid leitakse kõige sagedasemad kogu korpuse peale.

Kollokatsiooni koostamine on juhendites hästi kirjeldatud programmidel MonoconcEsy ja AntConc: esimesel on selgitatud sätteid ning AntConcil on juhis etappidena. MLCT juhendis on kirjeldatud lühidalt menüünuppude tähendusi, kuid kollokatsiooni menüü all on veel hulk valikuid, mille kohta juhendis täpsustav informatsioon puudub. Programmi juhend ei selgita, kuidas mingit funktsiooni kasutada, vaid kõigest suunab kasutajat õige menüü juurde.

[7] Kollokatsioon on sõnade koos esinemine: kollokatsiooni moodustavad sõnad, mis esinevad üksteise naabruses sagedamini, kui võiks eeldada nende eraldiesinemise sageduste põhjal (lk 7).

Kollokatsiooni koostamisel tuleb valida ulatus ehk piirid, kui kaugelt kollokatsioone otsitakse. MonoconcEsy võimaldab määrata märksõnast neli sõna kummaski suunas, AntConc 20 sõna, MLCT-s piirmäära ei ole. Koostamisel on AntConcis ja MLCT-s võimalik ise valida ka sobiv statistik ehk arvuline kirjeldaja, mille alusel kollokatsioonid leida: AntConcis on valikus MI ja T-skoor *(T-score)*, MLCT-s on erinevaid statistikuid kokku kümme. MonoconcEsys saab määrata ka eraldi stopp-sõnade loendi (*stop list)* sõnadest, mida kollokatsiooni koostamisel eirata.

Kollokatsiooni koostamist testisin programmides märksõna *üle* sisestamisel päringusse, ulatusega märksõnast üks sõna mõlemas suunas. Tabelis 3 on näha, et MLCT tulemus on teistest programmidest väga erinev ning seda põhjusel, et programm arvestab sagedasemad kollokatsioonid kogu korpuse peale. Seetõttu ei olnud võimalik proovida katset märksõnaga. Programmidel MonoconcEsy ja AntConc on neli sagedasemat kollokatsiooni samasugused, kuid esinemiste arv erineb sõnaga *on*. See tuleb sellest, et AntConc arvestab kokku eripidi kollokatsioonid, nt fraasi *on üle* hulka on AntConc lisanud ka *üle on*. Erinevused kollokatsioonides nende kahe programmi vahel tulevadki peamiselt põhjusel, et AntConc arvestab sellised variandid kokku, MonoconcEsy aga eraldi. AntConcis on sõna *miljoni* saanud kaks vastet, kuid MonoconcEsys mitte. Põhjus on selles, et AntConc jätab fraasis numbrid arvestamata: MonoconcEsys on olemas eraldi vasted *üle 50* ja *üle 17*, kuid AntConcis on sõna *miljoni* all koos *üle 50 miljoni* ja *üle 17 miljoni*.

Tabel 3. **Kollokatsiooni katse.**

Programmid	**Kollokatsioon sõnaga *üle* - esinemise hulk**				
MonoconcEsy	kahe - 3	veidi - 2	kolme - 2	vaatama - 2	on - 2
AntConc	on - 3	kahe - 3	veidi - 2	vaatama - 2	miljoni - 2
MLCT	ei ole - ?	ja siis - ?	ja kui - ?	ja ta - ?	ja ka - ?

Kollokatsiooni funktsiooni kasutamisel tuleb arvestada, et programmid MonoconcEsy ja AntConc võimaldavad otsida kollokatsioone märksõna järgi, MLCT aga moodustab neid kogu korpuse põhjal. MLCT-s on aga võimalik kollokatsiooni ulatust määrata

piiramatult ning statistikuid on rohkem kui teistel programmidel. MonoconcEsys saab lisaks tavapärasele kollokatsioonile koostada ka tabeli, mis näitab loenditena sagedasemaid kollokatsioone eri piirkondades *(Collocate frequency)*. MLCT-s ega MonoconcEsys ei saa kollokatsioonidel klikates näha seal kasutatud lauseid, mis on programmi üheks suureks miinuseks. AntConcis on see võimalik ning lisaks saab programmis tulemusi sorteerida.

4.4.4. N-grammid

Testitud programmidest kahes – AntConcis ja MLCT-s – on olemas n-grammi[8] funktsioon. AntConci juhendis on n-grammide koostamist kirjeldatud etappidena ning MLCT-s piirdutud menüüde tähenduse selgitamisega. Samas ei ole seda keeruline ka ilma juhendita koostada: tuleb valida soovitud n-gramm ning avaneb sageduse alusel koostatud loend. Juhenditest ei selgu, millise statistiku alusel n-grammid koostatakse, mistõttu võib arvata, et need on lihtsagedused. N-grammide koostamisel tuleb määrata ka nende soovitav pikkus: AntConcis on võimalikuks pikkuseks ühest kuni 99-ni, MLCT-s üheosalisest n-grammist kuni kuueosaliseni.

N-grammide katsel koostasin kaheosalised n-grammid ehk bigrammid. Vaadates tabelit 4, pole tulemused kahe programmi vahel väga erinevad, kuid MLCT on siiski jätnud mitmete fraaside hulka ühe või enam lauseid lisamata. Kuna MLCT-s ei saa loendite korral näha sinna arvestatud lauseid, siis tuleb kontrollimiseks koostada vastav konkordants ning leida nii laused, mis AntConcil on tulemuste hulgas, kuid MLCT-l mitte. Fraaside *ei ole*, *ja siis* ning *ei saa* tulemused erinevad suure algustähega variantide poolest, mis AntConci loendis on olemas, kuid MLCT-l puudu. *siis kui* fraasi esinemise hulk oli MLCT-s 19. Põhjuseks on see, et MLCT ei arvesta vastupidiselt AntConcile selliseid fraase, kus on sõnade vahel koma.

et ta tulemuseks oli MLCT-s 21, AntConcis 22. Kui aga panna MLCT konkordantsi vastav fraas, tuleb vastete hulgaks 22 nagu AntConcil bigrammides. Seetõttu ei saa kontrollida, miks ja mis lause MLCT bigrammides arvestamata on jäänud. Vaadates

[8] N-grammide funktsiooni kasutamisel saab leida üksteisele vahetult järgnevatest sõnadest moodustunud fraase (lk 9).

AntConci tulemusi, ei leidu seal ka mõnd sellist varianti, mida MLCT tuvastada ei suudaks.

Tabel 4. **N-grammi katse.**

Programmid	Bigramm - esinemise hulk				
AntConc	ei ole - 61	ja siis - 31	on ka - 30	siis kui - 30	ei saa - 29
MLCT	ei ole - 60	on ka - 30	ei saa - 28	ja siis - 28	et ta - 21

Katse põhjal selgus, et programmid esitavad n-gramme erinevatel alustel. AntConciga võrrelduna jätab MLCT lisamata suure algustähega variandid ning sellised, kus fraasis on sõnade vahel koma. Katse käigus ilmnes, et analüüsi muudab keerukamaks, kui programm (selles katses MLCT) ei võimalda näha lauseid, kus fraasid esinevad.

4.4.5. Muud funktsioonid

Märksõnaloendit saab töös testitud programmidest koostada AntConcis. AntConci juhendmaterjalis (Anthony 2014) on märksõnaloend defineeritud kui tööriist, millega on võimalik leida märksõnad, mis esinevad ühes korpuses sagedamini kui võrdluskorpuses *(reference corpus)*. See tähendab, et märksõnaloendi koostamiseks on tarvis teist korpust, millega uuritavat korpust võrrelda, et leida näiteks mingile kirjutusstiilile omaseid sõnu.

AntConci juhend kirjeldab märksõnaloendi koostamist etappidena nii, et kogu protsess on selgitatud ning loendi koostamine seda järgides kerge. Märksõnaloendi koostamisel on tarvis valida, kas tulemused arvutatakse välja log-tõepära *(log-likelihood)* või hii-ruudu funktsiooni *(chi squared)* põhjal, vaikimisi on selleks määratud esimene. Tulemust on võimalik sorteerida sageduse, tähestikuliselt märksõna esimese või viimase tähe järgi ning märksõna statistilise seose tugevuse põhjal *(Sort by Keyness).* Viimane näitab, millised sõnad on statistiliselt sellele tekstile iseloomulikud.

Erinevaid teksti analüüsi ja muutmise nuppe on programmil MLCT: näiteks on võimalik kaht listi omavahel võrrelda, kirjutatud mustri alusel tekstist sõnu eemaldada või filtreerida välja ühesuguseid lauseid. Lisaks saab programmis koostada korpust

internetiaadresside põhjal nagu ka TextSTAT-is, kuid mis jääb alla varem testitud korpuse koostamise programmidele, sest jätab kogu HTML-koodi korpusesse alles. Kuigi pealtnäha on programmis justkui palju eri funktsioone, on üsna raske mõista, kuidas neid kasutada.

Üldstatistilist informatsiooni korpuse kohta saab programmides MLCT, TextSTAT ja SCP. MLCT-s saab leida ridade ning sõnavormide arvu, TextSTAT-is ja SCP-s näha sõnade ja sõnavormide hulka korpuses. SCP-s on võimalik ka Yule'i K statistiku põhjal saada infot sõnavara rikkuse kohta. Yule'i K väärtus näitab, kui mitmekesine on kasutatud sõnavara – mida väiksem see arv on, seda laiem on järelikult sõnavara ehk kasutatakse rohkem erinevaid sõnu ja sõnevorme (Yule 1944: 83).

4.5. Tulemuste salvestamine

Funktsioonide kuvatavaid tulemusi on võimalik salvestada failidena. Kõigil programmidel on üheks salvestusformaadiks .txt, kuid võimalikud on ka muud formaadid (vt lisa 1). Eri formaadiga failides on sageli vormindus erinev. Näiteks programmi MonoconcEsy html-formaadis konkordantsi salvestus on visuaalselt parem kui txt-faili oma: märksõnad on esitatud värviliselt ning on selge vormindusega. AntConci html-failis on täpitähed puudu ning konkordantsi read järgnevad üksteisele. MLCT salvestatud tekstifaili vormindus oleneb programmist, millega see avada: kui Notepadis on konkordantsi tulemused esitatud jooksva tekstina, siis Microsoft Wordis avades on read üksteise all, märksõna eraldatult paremas ääres. TextSTAT-is on võimalik tulemus salvestada Microsoft Wordi või txt-failina. Kui salvestada .txt formaadis, on read ja märksõnad üksteise all joondatult, teisel juhul on märksõnade leidmine konteksti hulgast aeganõudvam. Simple Concordance Programil on kõigis formaatides vormistus samasugune nagu nende esitus programmis endas.

SCP-s, AntConcis ja MLCT-s on salvestamiskoht rippmenüüs nupu *File* all. AntConci ja MLCT juhendites on välja toodud salvestamisvõimalus- ning koht. SCP juhendis on

salvestamisest kirjutatud vaid sõnaloendi tutvustamise osas, mistõttu jääb mulje, et konkordantsi salvestada pole võimalik, kuigi see käib samamoodi *File* menüüst.

MonoconcEsys on igal funktsioonil salvestamiskoht eraldi vastavas menüüs. Protsessis aitab juhendmaterjal, sest see kirjeldab iga tulemuse salvestamist ning salvestamisaknas tehtavaid valikuid. TextSTAT-i juhendis ei ole salvestamisvõimalusest kirjutatud, kuid selleks on programmis eraldi rippmenüü *Export*, kus saab valida, millises formaadis konkordantsi või sõnaloendit salvestada.

Tulemuste salvestamine pole keeruline üheski programmis. MonoconcEsys on erinevaid valikuvõimalusi, millist informatsiooni faili juurde lisada, samas ei ole programmi .txt formaadis salvestus kergesti loetav. Salvestatud txt-failid olid kõige parema vormindusega programmidel SCP, AntConc ning TextSTAT.

4.6. Testimistulemuste kokkuvõte

Testisin programmides korpuse analüüsil läbitavaid protsesse ning analüüsifunktsioone. Testimistulemused tõid esile, et ideaalset programmi ei leidu: kõigi puhul saab välja tuua negatiivseid külgi kas kasutusmugavuses või funktsioneerimises. Programmi valimisel korpuse analüüsimiseks tuleb esmalt mõelda, milliseid analüüsifunktsioone on vaja kasutada. Enne programmi kasutamist oleks oluline arvesse võtta programmide probleemseid kohti, mida töös kirjeldanud olen, sest need võivad mõjutada analüüsile kuluvat aega ja tulemusi. Ülevaatlikku programmide kirjeldust tabeli kujul on võimalik näha lisas 1.

Viiest testitud programmist jäävad teistele veidi alla programmid SCP ja MLCT. SCP analüüsifunktsioonide katsetulemused olid küll sarnased AntConcile, kuid programm ei tööta *š*- ja *ž*-tähega ning puudub võimalus kasutada regulaaravaldisi. Kuna ülejäänud programmid töötavad korralikult kogu eesti tähestikuga, on SCP asemel kindlam kasutada mõnd teist vahendit. MLCT kasutajaliides on üsna segadust tekitav: korpust tuleb tihti uuesti laadida, pole võimalik eristada suur- ja väiketähti ning määrata, kas

soovitakse otsida sõna või sõnaosa. Samas saab programmis koostada mitmeosalisi n-gramme ning leidub muid väikseid analüüsivõimalusi.

Programmidega AntConc, MonoconcEsy ja TextSTAT ei teki analüüsimisel suuri probleeme, kui programmide vead on teada. AntConcil ilmnes testimise käigus kõige vähem probleeme, kuid programm ei sobi internetikeele analüüsimiseks, kus täpitähtede asemel kasutatakse sageli numbreid. AntConc ei arvesta kollokatsiooni osana numbreid (fraasi *üle 50* asemel loeb *üle 50 miljoni*) ja lisab eripidi fraasid kokku (kumbki ei pruugi olla alati miinus). Muus osas oli programmi lihtne ja mugav kasutada, sellel olid olemas kõik analüüsifunktsioonid ning juhendmaterjal toetab programmi kasutamist.

MonoconcEsy on küll sisuka juhendiga kasutajasõbralik programm, kuid kasutusmugavust vähendab see, et programm ei arvesta sõnu, mille küljes on mingi sümbol/kirjavahemärk. Eraldajate lisamine aitab, kuid alati ei pruugi kasutaja teada, millised sümbolid korpuses esinevad. Programmis SCP on küll võimalik koostada loend, mis näitab korpuses sisalduvaid sümboleid, kuid kontrollisin sümbolite esinemist eraldi *shelli* skriptiga ning tulemus oli programmi sümbolite loendist erinev. AntConc võimaldab kasutada samu funktsioone, tehes seega sama töö lihtsamini ja kiiremini. TextSTAT on programm, millega on hea teha lihtsat analüüsi, kasutades väikest ning märgendamata korpust. Programmi miinuseks on see, et konkordantsis ei ole märksõnad ega sorteeritud sõnad esitatud värviliselt. Kokkuvõttes jättis AntConc testitud programmidest kõige parema mulje ning oli kõige töökindlam. Programmi valik aga oleneb konkreetsetest vajadustest.

Kokkuvõte

Bakalaureusetöö andis ülevaate korpuse analüüsi programmidest ning nende sobivusest eestikeelse korpuse analüüsimiseks. Töös testisin programme AntConc, MonoconcEsy, Multilingual Corpus Toolkit, Simple Concordance Program ja TextSTAT, mis on tasuta kasutatavad ning töötavad Windowsi operatsioonisüsteemil. Testkorpuse loomisel testisin internetist korpuse koostamise programme BootCaT ja WebBootCat ning TextSTAT-is olevat korpuse koostamise funktsiooni. Töö põhirõhk oli siiski korpuse analüüsi, mitte koostamise tarkvara kirjeldamisel.

Korpuse koostamise programmide testimisel, kus tekstimaterjaliks olid Postimehe majandusuudised ning loomateemalised foorumid, selgus, et kõige terviklikuma ja korralikuma korpuse koostas programm WebBootCat. Programmi BootCaT korpuses puudus enamik foorumipostitusi ning TextSTAT-i koostatud korpuses oli palju kõrvalist teksti, näiteks internetilehekülje teiste uudiste pealkirjad. Seega sobib internetist korpuse kogumiseks programm WebBootCat, sest suudab teksti koguda erinevatest allikatest.

Korpuse analüüsi programmide testimine tõi esile, et probleeme esineb iga programmi kasutamisel, aga kui neist vigadest teadlik olla ning oma tööst lähtuvalt sobiv programm valida, on korpust nende abil võimalik edukalt analüüsida. Selgus, et viiest programmist võib parimaks pidada AntConci, millel on teistest programmidest rohkem analüüsi-funktsioone ja võimalusi. Eesti keele tähestikuga probleeme ei teki ning kasutamist toetab ka juhendmaterjal. Internetikeele analüüsimisel AntConciga võib aga tekkida vigu sõnadega, kus näiteks täpitähtede asemel on kirjutatud numbrid, sest programm poolitab sõnad numbrite juurest.

MonoconcEsy on samuti hästi funktsioneeriv ning kergesti kasutatav programm, kuid see ei arvesta analüüsimisel sõnu, mille küljes on mõni kirjavahemärk või mittetäheline sümbol. Selliste kirjavahemärkide ja sümbolite lisamine eraldajate loendisse küll aitab,

kuid alati ei pruugi kasutaja teada, millised sümbolid tema loodud korpuses esinevad. Programmi TextSTAT sobib kasutada lihtsamateks analüüsideks, sest kuigi programmil puudub hea visuaalne tulemuste esitus, on seda muus osas mugav kasutada. Simple Concordance Program oli testitutest ainus, kus esines probleem eesti tähestikuga: tähed *š* ja *ž* on programmi jaoks tundmatud. Kõige keerulisem oli kasutada programmi Multilingual Corpus Toolkit, millel on erinevaid päringuaknaid ja nuppe, mida on keeruline mõista ning kuna programmi juhendmaterjal on vaid suunav, ei toeta see piisavalt programmiga töötamist.

Seega tasuks eestikeelse korpuse analüüsimisel eelistada programmi AntConc, kuid sobivad on ka MonoconcEsy ja TextSTAT. Programmi valimisel tuleb seejuures arvestada korpuse iseärasustega ning sellega, milliseid analüüsimeetodeid on tarvis kasutada.

Kirjandus

Anagnostou, Nikolaos K., George R. S. Weir 2006. Review of software applications for deriving collocations. – ICTATLL Workshop 2006, 91–100.

Anthony, Laurence 2013. A critical look at software tools in corpus linguistics. – Linguistic Research 30, 141–161.

Baroni, Marco, Silvia Bernardini 2004. BootCaT: Bootstrapping corpora and terms from the web. – Proceedings of LREC 2004; http://bootcat.sslmit.unibo.it. Vaadatud 28.02.2016.

Baroni jt = Baroni, Marco, Adam Kilgarriff, Jan Pomikálek, Pavel Rychlý 2006. WebBootCat: a Web Tool for Instant Corpora. – In Proceeding of the EuraLex Conference. Italy: Edizioni dell'Orso s.r.l.

Bianchi, Francesca 2012. Culture, corpora and semantics. Methodological issues in using elicited and corpus data for cultural comparison. Salento: Coordinamento SIBA.

Biber jt = Biber, Douglas, Susan Conrad, Randi Reppen 1998. Corpus linguistics: Investigating language structure and use. Cambridge: Cambridge University Press.

Cheng, Winnie 2011. Exploring corpus linguistics: Language in action. Routledge.

Graham jt = Graham, Dorothy, Erik Van Veenendaal, Isabel Evans, Rex Black 2008. Foundations of Software Testing: ISTQB Certification. Cengage Learning EMEA.

Gries, Stefan Th. 2009. What is corpus linguistics? – Language and Linguistics Compass 3, 1225–1241.

Gries, Stefan Th., John Newman 2013. Creating and using corpora. – Research methods in linguistics. Ed. Robert J. Podesva, Devyani Sharma. Cambridge: Cambridge University Press, 257–287.

Kennedy, Graeme 1998. An Introduction to Corpus Linguistics. Studies in Language and Linguistics. Routledge.

Kilgarriff jt = Kilgarriff, Adam, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, Vít Suchomel 2014. The Sketch Engine: ten years on. – Lexicography 1, 7–36.

McEnery, Tony, Andrew Hardie 2012. Corpus Linguistics: Method, Theory and Practice. CambridgeTextbooks in Linguistics. Cambridge: Cambridge University Press.

Mitkov, Ruslan 2003. The Oxford Handbook of Computational Linguistics. Oxford: Oxford University Press.

Patton, R. 2005. Software Testing. Indianapolis: Sams Publishing.

Rayson, Paul E. 2002. Matrix: A Statistical Method and Software Tool for Linguistic Analysis through Corpus Comparison. – Ph.D. thesis. Lancaster: Lancaster University.

Scott, Mike 2012. Looking back or looking forward in corpus linguistics: What can the last 20 years suggest about the next? – Iberica 24, 75–85.

Wiechmann, Daniel, Stefan Fuhs 2006. Concordancing software. – Corpus Linguistics & Linguistic Theory 2(1), 107–127.

Yule, George Udny 1944. The Statistical Study of Literary Vocabulary. Cambridge: Cambridge University Press.

Programmide koduleheküljed ja dokumentatsioonid

Laurence Anthony's Website; http://www.laurenceanthony.net/software.html. Vaadatud 27.04.2016.

AdTAT; https://www.adelaide.edu.au/carst/resources-tools/adtat/. Vaadatud 27.04.2016.

All About Xaira; http://projects.oucs.ox.ac.uk/xaira/. Vaadatud 27.04.2016.

Anthony, Laurence 2014. AntConc (Windows, Macintosh OS X, and Linux). Build 3.4.4; http://www.laurenceanthony.net/software/antconc/releases/AntConc344/help.pdf. Vaadatud 27.04.2016.

Barlow, Michael 2015. MonoconcEsy: An Introduction to Concordancing. Houston: Athelstan.

BFSU PowerConc; http://www.bfsu-corpus.org/static/PowerConc.html. Vaadatud 27.04.2016.

BootCaT; http://bootcat.sslmit.unibo.it/. Vaadatud 28.02.2016.

ConcApp; http://concapp.software.informer.com/. Vaadatud 27.04.2016.

Hardie, Andrew 2016. CQPweb System Administrator's Manual; http://cwb.sourceforge.net/files/CQPwebAdminManual.pdf. Vaadatud 27.04.2016.

KH Coder; http://khc.sourceforge.net/en/. Vaadatud 27.04.2016.

LEXA; http://clu.uni.no/icame/lexainf.html. Vaadatud 27.04.2016.

MLCT; https://sites.google.com/site/scottpiaosite/software/mlct. Vaadatud 27.04.2016.

NoSketch Engine; https://nlp.fi.muni.cz/trac/noske/wiki/Downloads. Vaadatud 27.04.2016.

Piao, Scott Songlin 2009. User's Manual. – Multilingual Corpus Toolkit. Vaadatud 27.04.2016.

Simple Concordance Program; http://www.textworld.com/scp/. Vaadatud 27.04.2016.

TextSTAT; http://neon.niederlandistik.fu-berlin.de/en/textstat. Vaadatud 28.02.2016.

WebBootCat; https://www.sketchengine.co.uk/webbootcat/. Vaadatud 28.02.2016.

WordSmith Tools; http://lexically.net/wordsmith/. Vaadatud 27.04.2016.

WordStat 6; http://www.provalisresearch.com/Documents/WordStat6.pdf. Vaadatud 27.04.2016.

Testing Corpus Analysis Tools with the Estonian Language Corpus. Summary

The aim of this bachelor's thesis is to test corpus analysis tools also known as concordancers to compare them and focus on their suitability to work with the Estonian language corpora. Linguistic research nowadays bases often on an analysis of a text corpus. The aim of this thesis is to give overview of available programs and their suitability for researches, who have to create and analyse their own corpus. Current work will help users to find the appropriate program for their needs and guide them through the analysing process.

The thesis is divided into four sections. The first chapter gives an overview of corpus analysis tools and their development. The second part describes how the programs are selected for testing and which research methods are used. In the third chapter the tools for creating a web corpus are being tested. In the fourth chapter the test results of corpus analysis tools are presented.

Three web corpus compilation tools were tested: BootCat, WebBootCat and TextSTAT. The programs were tested using texts from the economic newspage of Postimees and web forums. Results showed that the most consistent corpus was created by WebBootCat. While the corpus of WebBootCat contained all the texts from webpages, the BootCat was unable to retrieve most of the forum posts. The corpus of TextSTAT included irrelevant text, e.g. headings of other news.

Five corpus analysis tools were examined and tested: AntConc, MonoconcEsy, Multilingual Corpus Toolkit, Simple Concordance Program and TextSTAT, which are free to use and run at least on Windows operation system. Test results revelead that problems occurred with all the programs, but if the user is aware of these pitfalls and chooses a program based on one's needs, the programs can be successfully utilized.

Based on the test results AntConc is the best tool. It has a variety of analysis possibilities and it works with the Estonian alphabet, whilst also having informative documentation. Working with Internet texts can cause some problems, e.g. words where numbers are used instead of the actual letters are split.

Also MonoconcEsy worked well and was easy to use. The biggest problem is that it does not recognize words, when some symbol (e.g. punctuation mark) is not separated with a space. This problem can be overcome by adding delimiters, but that assumes detecting all the occurring symbols in the corpus. TextSTAT is easy to use and is suitable for simple corpus analysises, but the visual presentation of results is poor. Simple Concordance Program was the only one not fully working with Estonian alphabet: it does not recognise the letters *š* nor *ž*. The most complicated program was Multingual Corpus Toolkit that has many search windows and buttons which are hard to understand and the documentation does not support working with the program.

In sum, AntConc is the most user friendly tool for analysing corpus in Estonian, but also MonoconcEsy and TextSTAT do their job well. When choosing a corpus analysis program it is advised to make clear the particular qualities of one's corpus and which analysis methods are needed.

Lisad

Lisa 1. Üldkokkuvõtlikud tabelid programmide funktsionaalsusest

	Simple Concordance Program	MonoconcEsy	AntConc	Multilingual Corpus Toolkit (MLCT)	TextSTAT
INSTALLIMINE	Installeerimisaken	Zip-fail	Allalaetav fail ongi töötav programm.	Zip-fail	Zip-fail
* Juhendmaterjal	Programmis Help nupu all.	Programmiga samas kaustas.	Programmis Help nupu all.	Programmis Help nupu all.	Puudub. Tuleb internetist otsida.
* Kodeering ja ühildumine eesti tähestikuga	ANSI	Unicode (UTF-16)	Unicode (UTF-8)	Unicode	Unicode
KORPUSE LAADIMINE JA KASUTAMINE					
Kuidas käib korpuse programmi laadimine?	Tuleb luua projekt, kuhu korpus laadida.	Iga kord tuleb korpus programmi uuesti laadida.	Iga kord tuleb korpus programmi uuesti laadida.	Iga kord tuleb korpus programmi uuesti laadida.	Tuleb luua projekt, kuhu korpus laadida.
Kas korpus võib olla nii ühes kui ka mitmes tekstifailis?	+	+	+	+	+
Milline võib olla korpuse formaat?	.ans; .asc; .txt	.txt	.txt; .html; .xml; .ant	.txt; .html; .xml; .sgml; .tex	MS Wordi failid (.doc; .docx); OpenOffice'i failid (.odt; .sxw); .txt; HTML failid; otse internetist materjal.
Kas korpust on võimalik vahepeal vahetada? (nt kasvõi faile juurde laadida/kustutada)	-	+	+	-	+
Kas märgendusi on võimalik peita?	+	+	+	-	-
PÄRINGU KOOSTAMINE JA TÖÖTLEMINE					
Kas on võimalik kasutada metamärke ja regulaaravaldisi?	-	+	+	+	+
Kas programm eristab suur- ja väiketähti?	+	+	+	+	+
Kas saab määrata sätetes, et otsitaks i seseisvat sõna (et tulemuste hulgas poleks sõnaosi)? Või vastupidi?	+	-	+	-	+
Kas päringutulemuste hulka saab suurendada-vähendada?	-	+	-	-	-

	Simple Concordance Program	MonoconcEsy	AntConc	Multilingual Corpus Toolkit (MLCT)	TextSTAT
FUNKTSIOONID					
Millised funktsioonid on programmil?	Konkordants, sõnaloend	Konkordants, sõnaloend, kollokatsioon	Konkordants, sõnaloend, kollokatsioon, n-grammid, märksõnaloend	Konkordants, sõnaloend, kollokatsioon, n-grammid	Konkordants, sõnaloend
*** Konkordants**	+	+	+	+	+
Kas saab muuta konteksti hulka märksõna ümber?	+	+	+	+	+
Kuidas konkordantsi päringutulemust sorteerida saab?	Korpuses esinemise järjekorra järgi, tähestikuliselt vasakule-paremale jääva sõna järgi	2 sõna vasakule-paremale	20 sõna vasakule-paremale	3 sõna vasakule-paremale	Korpuses esinemise järjekorra järgi, tähestikuliselt vasakule-paremale jääva sõna järgi
Kas on olemas tasemeline sorteerimine?	-	+	+	-	-
Kas päringutulemuste aknas on võimalik märksõnal klikates näha seda kohta ka failis?	-	+	+	-	+
*** Sõnaloend**	+	+	+	+	+
Kuidas saab sõnaloendit sorteerida?	Tähestikuliselt, sageduse ja sõna pikkuse järgi	Tähestikuliselt, sageduse järgi	Tähestikuliselt, sageduse järgi	Tähestikuliselt, sageduse järgi	Tähestikuliselt, sageduse järgi
Milliseid loendeid on lisaks sõnaloendile võimalik teha?	Täheloend, sümbolite loend. Sõnade sageduse määr üldise korpuse sagedusega võrreldes	Kollokatsiooni loend	-	Lausete pikkuse loend	-
*** Kollokatsioon**	PUUDUB	+	+	+	PUUDUB
Milline on kollokatsiooni ulatus?		4 sõna vasakule-paremale	20 sõna vasakule-paremale	Piiramatu	
*** N-grammid**	PUUDUB	PUUDUB	+	+	PUUDUB
Millise pikkusega n-gramme on võimalik koostada?			1 kuni 99	1 kuni 6	
*** Märksõnaloend**	PUUDUB	PUUDUB	+	PUUDUB	PUUDUB
Kuidas saab märksõnaloendit sorteerida?			Tähestikuliselt, sageduse ja märksõna statistilise seose tugevuse järgi		
*** Muud funktsioonid**	Võimalik leida sõnade ja sõnavormide hulk, sõnavara rikkuse kohta	-	-	Internetist korpuse kogumine, võimalik leida ridade ja sõnavormide hulk	Internetist korpuse kogumine, võimalik leida sõnade ja sõnavormide hulk
TULEMUSTE SALVESTAMINE					
Milline peab olema väljundi formaat?	.txt; .rtf; .htm	.txt; .html	.txt; .html; .xml; .ant	Erinevad formaadid	.txt; MS Word; .CSV; MS Excel

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Helen Kaljumäe, sündinud 21.04.1994,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Korpuse analüüsi programmide testimine eestikeelse korpuse põhjal“, mille juhendajad on Kadri Muischnek ja Kristel Uiboaed,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 26. mai 2016.